PART II
RESEARCH PAPERS

4. MOBILITY AND BENCHMARKING

Features of Intelligence Required by Unmanned Ground Vehicles

James S. Albus
National Institute of Standards and Technology
Gaithersburg, MD 20899

ABSTRACT

A definition of intelligence is given in terms of performance that can be quantitatively measured. Behaviors required of unmanned ground vehicles are described and computational requirements for intelligent control at seven hierarchical levels in a military scout platoon are outlined. Metrics and measurements are suggested for evaluating the performance of unmanned ground vehicles. Calibrated data and test facilities are suggested to facilitate the development of intelligent systems.

KEYWORDS: intelligence, intelligent systems, unmanned ground vehicles, scout platoon, autonomous vehicles, metrics, measures

1. DEFINITIONS

The definition of intelligence is a controversial subject. Hardly any two persons define intelligence the same. Some even question whether intelligence can be defined at all. Yet, if we are to perform serious research on intelligent systems, we must not only be able to define intelligence, we must be able to quantitatively measure it. Thus, for the purpose of discussion of the issues addressed in this paper, we will define intelligence as follows [1]:

Df: **intelligence**

the ability to act appropriately in an uncertain environment

Df: **appropriate action**

that which maximizes the probability of success

Df: **success**

the achievement or maintenance of behavioral goals

Df: **behavioral goal**

a desired state of the environment that a behavior is designed to achieve or maintain

This definition of intelligence addresses both biological and machine embodiments. It admits a broad spectrum of behaviors, from the simple to the complex. We deliberately do not define intelligence in binary terms (i.e., this machine is intelligent and this one is not, or this species is intelligent and this one is not) and we do not limit our definition of intelligence to behavior that is beyond our understanding. Our definition includes the entire spectrum of intellectual capabilities from that of a paramecium to that of an Einstein, from that of a thermostat to that of the most sophisticated computer system. We include the ability of a robot to spot-weld an automobile body, the ability of a bee to navigate in a field of wild flowers, a squirrel to jump from limb to limb, a duck to land in a high wind, and a swallow to catch insects in flight above a field of wild flowers. We include the ability of blue jays to battle in the bushes for a nesting site, a pride of lions to conduct a coordinated attack on a wildebeest, and a flock of geese to migrate south for the winter. We include a human’s ability to bake a cake, play the violin, read a book, write a poem, fight a war, or invent a computer.

Our definition of intelligence recognizes degrees, or levels, of intelligence. These are determined by the following parameters: 1) the computational power and memory capacity of the system's brain (or computer), 2) the sophistication of the processes the system employs for sensory processing, world modeling, behavior generation, value judgment, and communication, and 3) the quality and quantity of information and values the system has stored in its memory. The measure of intelligence is success in solving problems, anticipating the future, and acting so as to maximize the likelihood of achieving goals. Success can be measured by various criteria of performance (including life or death, pain or pleasure, reliability in goal achievement, cost in time and resources, and others.) Different levels of intelligence produce different probabilities of success.

Our definition of intelligence also has many dimensions. For example, the ability to understand what is visually perceived is qualitatively different from the ability to comprehend what is spoken. The ability to reason about mathematics and logic lies along a different dimension from the ability to compose music and verse. The ability to choose wisely involves both the ability to predict the future and the ability to accurately assess the cost or benefit of predicted future states. Along each of these dimensions, there exists a continuum. Thus, the space of intelligent systems is a multidimensional continuum wherein non-intelligent systems occupy a point at the origin.

At a minimum, intelligence requires the ability to sense the environment, to make decisions, and to control action. Higher levels of intelligence may include the ability to recognize objects and events, to represent knowledge in a world model, and to reason about and plan for the future. In advanced forms, intelligence provides the capacity to predict the future, to perceive and understand what is going on in the world, to choose wisely, and to act successfully under a large variety of circumstances so as to survive, prosper, and replicate in a complex, competitive, and often hostile environment.

From the viewpoint of control theory, intelligence might be defined as a knowledgeable "helmsman of behavior." Intelligence is a phenomenon which emerges as a result of the integration of knowledge and feedback into a sensory-interactive, goal-directed control system that can make plans and generate effective purposeful action to achieve goals.

From the viewpoint of psychology or biology, intelligence might be defined as a behavioral strategy that gives each individual a means for maximizing the likelihood of success in achieving its goals in an uncertain and often hostile environment. Intelligence results from the integration of perception, reason, emotion, and behavior in a sensing, perceiving, knowing, feeling, caring, planning, and acting system that can formulate and achieve goals.

2. REQUIREMENTS FOR UNMANNED GROUND VEHICLES

The features of intelligence required by an Unmanned Ground Vehicle (UGV) depends on many factors, such as:

What does the UGV have to do?

Does it simply wander through a lab looking for soft drink cans?

Does it have to operate outside? Travel long distances? Perform difficult tasks?

How complex and uncertain is the environment?

Where is it expected to operate? On well marked roads? On unmarked roads? Gravel or dirt roads? Roads grown up with weeds and brush? Off roads? In tall grass and weeds? In woods? Does it have to cross streams? Are there bridges or fords available? What kind of maps are available? How accurate are they? How recent?

How dynamic and hostile is the environment?

Are there moving obstacles? What are the lighting conditions? Are obstacles located above or below ground level? Are there other agents competing for the goal? Are there enemy agents with deadly weapons?

What are costs, risks, and benefits?

What are the stakes? Life or death? Win or lose?

What are goals?

Attack? Defend? Escape? Detect and track enemy targets? Remain undetected?

What are tasks?

Pick up an object? Use a tool? Dig a ditch? Cross a stream? Establish an observation post? Discover an enemy vehicle? Analyze enemy behavior? Identify a face in a crowd?

What sensors are available?

CCD cameras? FLIRs? LADARs? Radars? Sonars? Inertial? GPS? Beacons? Reflectors? Tactile? Force? Encoders?

What actuators are to be controlled?

Manipulators? Grippers? Power train? Legs or Wheels? Steering? Brakes? Switches?

How much is known apriori?

Maps? Lists of objects and their attributes? State of objects? Behavior of objects? Rules?

What skills and abilities are required?

Locomotion? Manipulation? Perception? Communication? Reasoning? Speech understanding? Written text understanding? In what languages?

The above questions are so open ended that it is futile to try to address all these issues simultaneously. To focus our efforts, we select an example of a problem that is difficult enough to be challenging, well defined enough to quantitatively measure performance, easy enough that it probably can be achieved using available technology, and useful enough that it is worth spending time and resources to solve it. The problem that we have selected it that of an unmanned ground vehicle for military scout operations.

3. A SCOUT PLATOON EXAMPLE

To illustrate the types of issues that will be addressed, an example is given below of a seven level hierarchy for a scout platoon attached to a battalion. The specific numbers and functions described in this example are illustrative only. They are meant only to illustrate how the generic structure and function of an intelligent system might be instantiated in the 4D/RCS architecture [2] designed for the Army's Demo III experimental unmanned ground vehicle program. [3]

Exact numbers for the actual system are still under development.

Level 7 -- Battalion

An armored battalion is a unit that consists of a group of M1 or Bradley companies and a scout platoon. A computational node at level 7 of the 4D/RCS architecture corresponds to a battalion headquarters unit, consisting of a battalion commander, several company commanders, a scout platoon leader, and support staff. (In principle, any or all of these could be humans or intelligent agent software processes. In practice, they are all humans.)

The battalion headquarters unit plans activities and allocates resources for the armored companies and the scout platoon attached to the battalion. Incoming orders to the battalion are decomposed by the battalion commander into assignments for the companies and the scout platoon. Resources and assets are allocated to each subordinate unit, and a schedule is generated for each unit to maneuver and carry out assigned operations. Together, these assignments, allocations, and schedules comprise a plan. The plan may be devised by the battalion commander alone, or in consultation with his subordinate unit leaders. The battalion level planning process may consider the exposure of each unit's movements to enemy observation, and the traversability of roads and cross-country routes. The battalion commander typically defines the rules of engagement for the units under his command and works with his unit leaders to develop a schedule that meets the objectives of the mission orders given to the battalion. In the 4-D/RCS battalion node, plans are computed for a period of about 24 hours(h) and recomputed at least once every 2 h, or more often if necessary. Desired positions for each of the subordinate units at about 2 h intervals are computed.

The 4D/RCS architecture provides a surrogate battalion node in each individual vehicle to perform the functions of the battalion headquarters unit when the vehicle is not in direct communication with its chain of command. The surrogate node plans activities for the vehicle on a battalion level time scale and estimates what platoon and section level operations should be executed to follow that plan. The surrogate battalion node considers the exposure of scout platoon operations to enemy observations, and the traversability of roads and cross-country routes.

In the surrogate battalion node in each vehicle, the 4-D/RCS world model maintains a knowledge database containing a copy of the battalion level knowledge database that is relevant to that vehicle. It contains names and attributes of friendly and enemy forces and of the force levels required to engage them. Maps have a range of 1000 km (i.e. more than the distance that a vehicle is likely to travel in a 24 h day at a Demo III speed of 36 km per hour (10 m/s)) with a resolution of about 400 m. Maps describe the terrain and location of friendly and enemy forces (to the extent that they are known), and roads, bridges, towns, and obstacles such as mountains, rivers, and woods. Battalion level maps may be updated from intelligence reports.

4-D/RCS sensory processing in the surrogate battalion node integrates information about the movement of forces, the level of supplies, and the operational status of all the units in the battalion, plus intelligence about enemy units in the area of concern to the company. This information is used to update maps and lists in the knowledge database so as to keep it accurate and current.

The surrogate battalion node also contains value judgment functions (e.g., calculating the risk of casualties) that enable the battalion commander to evaluate the cost and benefit of various tactical options. To the extent that the knowledge, skills, and abilities in the surrogate battalion node is identical with that in the real battalion node, the surrogate battalion node will make the same decisions as the real battalion headquarters node.

An operator interface allows human operators (either on-site or remotely) to visualize information such as the deployment and movement of forces, the availability of ammunition, and the overall situation within the scope of attention of the battalion commander. The operator can intervene to change priorities, alter tactics, or redirect the allocation of resources.

Output from the battalion level through the company commanders and scout platoon leader comprise input commands to the company/platoon level. Armor company commanders and the scout platoon leader are expected to issue commands to their respective units, monitor how well their units are following the battalion plan, and make adjustments as necessary to keep on plan. New output commands may be issued at any time, and typically consist of tasks expected to require about 2 h to complete.

Level 6—Platoon

A scout platoon is a unit that typically consists of ten HMMWVs or Bradley vehicles organized into one or more sections. For the Demo III project, a scout platoon will consist of six manned HMMWVs and four UGVs. A 4-D/RCS node at the Platoon level corresponds to a scout platoon headquarters unit. It consists of a platoon commander plus his/her section leaders. (Any of these could be humans or intelligent agent software processes, in any combination.) The platoon commander and section leaders plan activities and allocate resources for the sections in the platoon. Platoon orders are decomposed into job assignments for each section. Resources are allocated, and a schedule of activities is generated for each section. Movements are planned relative to major terrain features and other sections within the platoon. Inter-section formations are selected on the basis of tactical goals, stealth requirements, and other priorities. At the platoon level, plans are computed for a period of about 2 h into the future, and replanning is done about every 10 min, or more often if necessary. Section waypoints about 10 min apart are computed.

The surrogate platoon node in each vehicle performs the functions of the platoon headquarters unit when the vehicle is not in direct communication with the chain of command. It plans activities for the vehicle on a platoon level time scale and estimates what vehicle level maneuvers should be executed in order to follow that plan. Movements are planned relative to major terrain features and other vehicles within the platoon.

At the platoon level, the 4-D/RCS world model symbolic database contains names and attributes of targets, and the weapons and ammunition necessary to attack them. Maps with a range of about 100 km (i.e. more than the distance a platoon is likely to travel in 2 h) and resolution of about 40 m describe the location of objectives, and routing between them. Sensory processing integrates intelligence about the location and status of friendly and enemy forces. Value judgment evaluates tactical options for achieving section objectives. An operator interface allows human operators to visualize the status of operations and the movement of vehicles within the section formation. Operators can intervene to change priorities and reorder the plan of operations. Section leaders are expected to sequence commands to their respective sections, monitor how well their sections are following the platoon plan, and make adjustments as necessary to keep on plan. The output from the platoon level through the section leaders are commands issued to sections to perform maneuvers and engage enemy units in particular sectors of the battlefield. Output commands may be issued at any time, but typically are planned to change only about once every 5 min.

Level 5—Section

A scout section is a unit that consists of a group of individual scout vehicles such as HMMWVs and UGVs. A 4-D/RCS node at the section level corresponds to a section leader and vehicle commanders (humans or intelligent software agents). The section leader assigns duties to the vehicles in his section and coordinates the vehicle commanders in scheduling cooperative activities of the vehicles within a section. Orders are decomposed into assignments for each vehicle, and a schedule is developed for each vehicle to maneuver in formation within assigned corridors taking advantage of local terrain features and avoiding obstacles. Plans are developed to conduct coordinated maneuvers and to perform reconnaissance, surveillance, or target acquisition functions. At the section level, plans are computed for about 10 min into the future, and replanning is done about every 1 min, or more often if necessary. Vehicle waypoints about 1 min apart are computed.

The surrogate section node in each UGV performs the functions of the section command unit when the UGV is not in direct communication with the section commander. The surrogate node plans activities for the UGV on a section level time scale and estimates what vehicle level maneuvers should be executed in order to follow that plan.

At the section level, the 4-D/RCS world model symbolic database contains names, coordinates, and other attributes of other vehicles within the section, other sections, and potential enemy targets. Maps with a range of about 10 km and a resolution of about 30 m are typical. Maps at the section level describe the location of vehicles, targets, landmarks, and local terrain features such as buildings, roads, woods, fields, streams, fences, ponds, etc. Sensory processing determines the position of landmarks and terrain features, and tracks the motion of groups of vehicles and targets. Value judgment evaluates plans and computes cost, risk, and payoff of various alternatives. An operator interface allows human operators to visualize the status of the battlefield within the scope of the section, or to intervene to change priorities and reorder the sequence of operations or selection of targets. Vehicle commanders issue commands to their respective vehicles, monitor how well plans are being followed, and make adjustments as necessary to keep on plan. Output commands to individual vehicles to engage targets or maneuver relative to landmarks or other vehicles may be issued at any time, but on average are planned for tasks that last about 1 min.

Level 4—Individual vehicle

The vehicle is a unit that consists of a group of subsystems, such as locomotion, attention, communication, and mission package. A manned scout vehicle may have a driver, vehicle commander, and a lookout. Thus, a 4-D/RCS node at the vehicle level corresponds to a vehicle commander plus subsystem planners and executors. The vehicle commander assigns jobs to subsystems and schedules the activities of all the subsystems within the vehicle. A schedule of waypoints is developed by the locomotion subsystem to avoid obstacles, maintain position relative to nearby vehicles, and achieve desired vehicle heading and speed along the desired path on roads or cross-country. A schedule of tracking activities is generated for the attention subsystem to track obstacles, other vehicles, and targets. A schedule of activities is generated for the mission package and the communication subsystems. Waypoints and task activities about 5 s apart out to a planning horizon of 1 min are replanned every 5 s, or more often if necessary.

At the vehicle level, the world model symbolic database contains names (identifiers) and attributes of objects -- for example, the size, shape, and surface characteristics of roads, ground cover, or objects such as rocks, trees, bushes, mud, and water. Maps are generated from on-board sensors with a range of about 500 m and resolution of 4 meters. These maps are registered and overlaid with 40 meter resolution data from Section level maps. Maps represent object positions (relative to the vehicle) and dimensions of road surfaces, buildings, trees, craters, and ditches. Sensory processing measures object dimensions and distances, and computes relative motion. Value judgment evaluates trajectory planning and sensor dwell time sequences. An operator interface allows a human operator to visualize the status of operations of the vehicle, and to intervene to change priorities or steer the vehicle through difficult situations. Subsystem controller executors sequence commands to subsystems, monitor how well plans are being followed and modify parameters as necessary to keep on plan. Output commands to subsystems may be issued at any time, but typically are planned to change only about once every 5 s.

Level 3—Subsystem level

Each subsystem node is a unit consisting of a controller for a group of related Primitive level systems such as Primitive mobility, Gaze control, Communication, and Mission package sub-subsystems. A 4-D/RCS node at the Subsystem Level assigns jobs to each of its Primitive sub-subsystems and coordinates the activities among them. A schedule of Primitive mobility waypoints and Primitive mobility actions is developed to avoid obstacles. A schedule of pointing commands is generated for aiming cameras and sensors. A schedule of messages is generated for communications, and a schedule of actions is developed for operating the mission package sub-subsystems. The Primitive mobility way points are about 500 ms apart out to a planning horizon of about 5 s in the future. A new plan is generated about every 500 ms.

At the Subsystem level, the world model symbolic database contains names and attributes of environmental features such as road edges, holes, obstacles, ditches, and targets. Vehicle centered maps with a range of 50 meters and resolution of 40 cm are generated using data from range sensors. These maps represent the shape and location of terrain features and obstacle boundaries. The Demo III LADAR and stereo cameras measure position and range (out to about 50 m) of surfaces in the environment. Sensory processing computes surface properties such as dimensions, area, orientation, texture, and motion. Value judgment supports planning of steering and aiming computations, and evaluates sensor data quality. An operator interface allows a human operator to visualize the state of the vehicle, or to intervene to change mode or interrupt the sequence of operations. Subsystem executors compute at a 5 Hz clock rate. They sequence commands to primitive systems, monitor how well plans are being followed, and modify parameters as necessary to keep on plan. Output commands to Primitive sub-subsystems may be issued at any 200 ms interval, but typically are planned to change on average about once every 500 ms.

Level 2— Primitive level

Each node at the primitive level is a unit consisting of a group of controllers that plan and execute velocities and accelerations to optimize dynamic performance of components such as steering, braking, acceleration, gear shift, camera pointing, and weapon loading and pointing, taking into consideration dynamical interaction between mass, stiffness, force, and time. Communication messages are encoded into words and strings of symbols. Velocity and acceleration set points are planned every 50 ms out to a planning horizon of 500 ms.

The world model symbolic database contains names and attributes of state variables and features such as target trajectories and edges of objects. Maps are generated from camera data. Five meter maps have a resolution of about 4 cm. Driving plans can be represented by predicted tire tracks on the map, and visual attention plans by predicted fixation points in the visual field.

Sensory processing computes linear image features such as occluding edges, boundaries, and vertices and detects strings of events. Value judgment cost functions support dynamic trajectory optimization. An operator interface allows a human operator to visualize the state of each controller, and to intervene to change mode or override velocities. Primitive level executors keep track of how well plans are being followed, and modify parameters as necessary to keep within tolerance. Primitive executors compute at a 20 Hz clock rate. Output commands are issued to the Servo level to adjust set points for vehicle steering, velocity, and acceleration or for pointing sensors or weapons platforms. Output commands are issued every 50 ms.

Level 1—Servo level

Each node at the servo level is a unit consisting of a group of controllers that plan and execute actuator motions and forces, and generate discrete outputs. Communication message bit streams are produced. The servo level transforms commands from component to actuator coordinates and computes motion or torque commands for each actuator. Desired forces, velocities, and discrete outputs are planned for 20 ms intervals out to a planning horizon of 50 ms.

The world model symbolic database contains values of state variables such as actuator positions, velocities, and forces, pressure sensor readings, position of switches, and gear shift settings. Sensory processing detects events, and scales and filters data from individual sensors that measure position, velocity, force, torque, and pressure. Sensory processing also computes pixel attributes in images such as spatial and temporal gradients, stereo disparity, range, color, and image flow. An operator interface allows a human operator to visualize the state of the machine, or to intervene to change mode, set switches, or jog individual actuators. Executors servo actuators and motors to follow planned trajectories. Position, velocity, or force servoing may be implemented, and in various combinations. Servo executors compute at a 200 Hz clock rate. Motion output commands to power amplifiers specify desired actuator torque or power every 5 ms. Discrete output commands produce switch closures and activate relays and solenoids.

The above example illustrates how the 4-D/RCS multilevel hierarchical architecture assigns different responsibilities and duties to various levels of the hierarchy with different range and resolution in time and space at each level. At each level, sensory data is processed, entities are recognized, world model representations are maintained, and tasks are decomposed into parallel and sequential subtasks, to be performed by cooperating sets of agents. At each level, feedback from sensors reactively closes a control loop allowing each agent to respond and react to unexpected events.

At each level, there is a characteristic range and resolution in space and time, a characteristic bandwidth and response time, and a characteristic planning horizon and level of detail in plans. The 4-D/RCS architecture thus organizes the planning of behavior, the control of action, and the focusing of computational resources such that functional processes at each level have a limited amount of responsibility and a manageable level of complexity.

4. DEMO III CONTROL HIERARCHY

Figure 1 is a high-level block diagram of the first five levels in the 4-D/RCS architecture for Demo III. On the right, Behavior Generation modules decompose high level mission commands into low level actions. The text beside the Planner and Executor at each level indicates the planning horizon, replanning rate, and reaction latency, and the rate at which new commands are typically generated at each level. Each planner has a world model simulator that is appropriate for the problems encountered at its level.

In the center of Figure 1, each map as a range and resolution that is appropriate for path planning at its level. At each level, there are symbolic data structures and segmented images with labeled regions that describe entities, events, and situations that are relevant to decisions that must be made at that level. On the left is a sensory processing hierarchy that extracts information from the sensory data stream that is needed to keep the world model knowledge database current and accurate.

At the bottom of Figure 1 are actuators that act on the world and sensors that measure phenomena in the world. The Demo III vehicles will have a variety of sensors including a laser range imager (LADAR), stereo CCD (charge coupled device) cameras, stereo forward looking infra red (FLIR) devices, a color CCD, a vegetation penetrating radar, GPS (Global Positioning System), an inertial navigation package, actuator feedback sensors, and a variety of internal sensors for measuring parameters such as engine temperature, speed, vibration, oil pressure, and fuel level. The vehicle also will carry a Reconnaissance, Surveillance, and Target Acquisition (RSTA) mission package that will include long-range cameras and FLIRs, a laser range finder, and an acoustic package.

In Figure 1, the bottom (Servo) level has no map representation. The Servo level deals with actuator dynamics and reacts to sensory feedback from actuator sensors. The Primitive level map has range of 5 m with resolution of 4 cm. This enables the vehicle to make small path corrections to avoid bumps and ruts during the 500 ms planning horizon of the Primitive level. The Primitive level also uses accelerometer data to control vehicle dynamics and prevent roll-over during high speed driving.

The Subsystem level map has range of 50 m with resolution of 40 cm. This map is used to plan about 5 s into the future to find a path that avoids obstacles and provides a smooth and efficient ride. The Vehicle level map has a range of 500 m with resolution of 4 m. This map is used to plan paths about 1 min into the future taking into account terrain features such as roads, bushes, gullies, or tree lines. The Section level map has a range of 5000 m with resolution of about 40 m. This map is used to plan about 10 m into the future to accomplish tactical behaviors. Higher level maps (not shown in Figure 1) are used to plan section and platoon missions lasting about 2 and 24 h respectively. These are derived from military maps and intelligence provided by the digital battlefield database.

4D/RCS planners are designed to generate new plans well before current plans become obsolete. Thus, action can always take place in the context of a recent plan, and feedback through the executors can close reactive control loops using recently selected control parameters. To meet the demands of Demo III, the 4D/RCS architecture specifies that replanning should occur within about one-tenth of the planning horizon at each level (e.g., replanning at the Vehicle level will occur about every 5 s.)

Executors can react to sensory feedback even faster (e.g., reaction at the Vehicle level will occur within 500 ms). If the Executor senses an error between its output CommandGoal and the predicted state (status from the subordinate BG Planner) at the GoalTime, it may react by modifying the commanded action so as to cope with that error. This closes a feedback loop through the Executor at that level within the specified reaction latency.

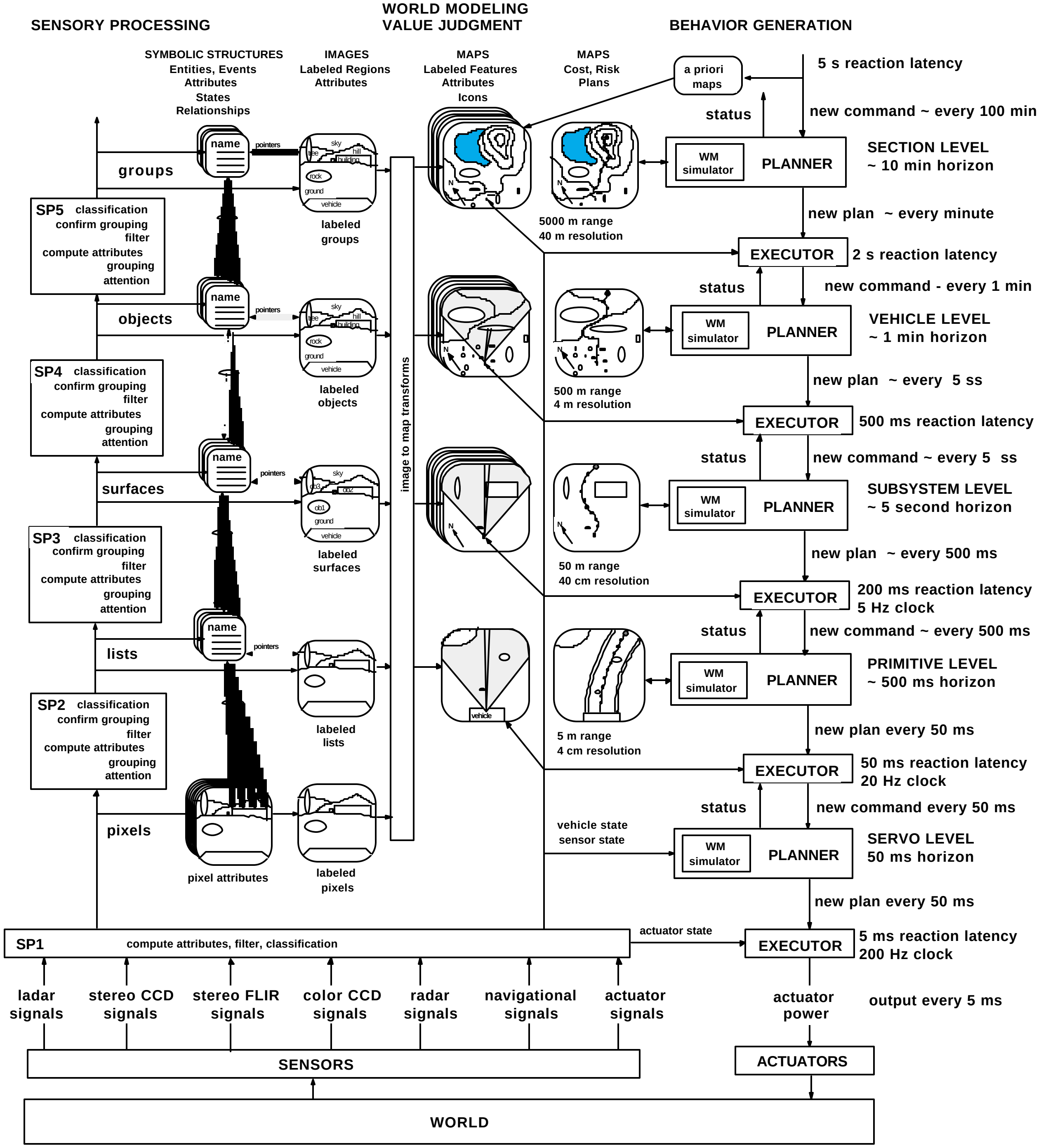


Figure 1. Five levels of the 4-D/RCS architecture. On the right are Planner and Executor modules. In the center are maps for representing terrain features, road, bridges, vehicles, friendly/enemy positions, and the cost and risk of traversing various regions. On the left are Sensory Processing functions, symbolic representations of entities and events, and segmented images with labeled regions.

The type of Executor reaction depends on the size and nature of the detected error. If the error is small, the Executor may simply modify its CommandedAction in a manner designed to reduce the error. For example, if the status reported from the subordinate planner indicates that the vehicle is going to arrive at the goal point late, the Executor might modify its CommandedAction to speed up or delete some low priority activities. However, if the error is out of range, the Executor may select a stored emergency plan from an exception handler, substitute it for the current plan, and modify its CommandedAction and CommandGoal to its subordinate planner appropriately. For example, an event such as the discovery of an unexpected obstacle in the AM planned path (generated by the Vehicle Planner) may cause the AM planner to make a plan that deviates significantly from its commanded goal. In this case, the Vehicle level Executor may modify its CommandedAction in a manner designed to buy time for the Vehicle level Planner to generate a new AM plan. For example, it may command the AM level to reduce speed or stop and direct AM driving cameras or RSTA sensors to collect information about the obstacle while a new AM plan is being generated by the Vehicle level planner. All of this Executor response should take place within the 500 ms reaction latency of the Vehicle level Executor.

Typically, evoking an emergency plan will cause the Executor to request its Planner to immediately begin a new replanning cycle. As shown in Figure 1, the period required for replanning at the Vehicle level is 5 s. The replanning period at the AM level is 0.5 s. Thus, the emergency plan evoked by the Vehicle level Executor can handle the problem of what the AM level should plan to do over the next 5 s while the Vehicle level planner generates a new AM plan out to its 1 min planning horizon.

5. GENERIC BEHAVIORS OF SCOUT VEHICLES

Navigate from A to B

Point A may be several km from point B. What kind of roads are available? How much traffic will be present? A scout vehicle may be required to stay off of roads, to maneuver through hilly fields and woods, and cope with fences, washes, and streams.

Avoid obstacles

The simplest obstacles are those that stick up from flat ground and are not obscured by foliage. The most difficult are ditches that are obscured by foliage. It is important to be able to distinguish grass and weeds that the vehicle can drive through from grass and weeds that conceal obstacles. In some cases, the only way to tell the difference is to drive slowly and stop when the vehicle encounters stiff resistance, or when the front wheels drop over the edge of a ditch, or sink into the mud.

Compute terrain attributes and classify terrain features

The first requirement is to map the terrain geometry and topology. The second is compute attributes such as color, texture, slope, size, and shape of regions of terrain. The third is to compare attributes of terrain regions with class attributes so as to classify terrain regions as road, dirt, grass, rocks, brush, trees, and bogs.

Drive autonomously

Driving autonomously covers a wide range of situations. Driving on an empty freeway is quite different from driving in downtown Istanbul. Driving with traffic on a freeway requires the ability to recognize lane markings, detect and track other vehicles, detect and avoid obstacles in the roadway, and obey road signs.

Driving at normal human speeds on narrow roads and cross country is more difficult. Road edges may be poorly defined and lane markings often do not exist. There may be bumps or ditches that will damage the vehicle if struck at high speeds.

Autonomous driving in suburban or downtown streets requires the ability to detect and predict the behavior of pedestrians, other vehicles, to read road signs, and respond to traffic signals, including hand signals from humans.

In driving cross country, there is no guarantee that a chosen path is even feasible. There may be hidden obstacles such as ditches, streams, fences, hills, brush, or woods that are impassable. The vehicle must be able to back up, and try alternate routes when the planned path is blocked.

Classify landmarks, objects, places, and situations

It is easy to get lost. GPS is not always available. Critical path waypoints may not appear on a map, or may be incorrectly represented. The unexpected appearance of an enemy may require immediate action. The ability to recognize a likely spot for an enemy sniper in time to take evasive action may be critical to survival.

Recognize and track other vehicles, avoid collisions

On-coming traffic on narrow roads is a major problem. One must drive very close to oncoming vehicles to stay on the road. One must estimate whether the oncoming vehicle is in its own lane on its own side of the road, and whether there is room on the road for two vehicles to safely pass. To do that one must detect the road edges at a great distance and measure the relative position of the on-coming vehicle between the road edges. There is very little margin for error in space or time.

Predict behavior of pedestrians and other vehicles in traffic

Driving in traffic requires the self vehicle to not only detect, but to predict where pedestrians and other vehicles will be in the future. For example, on a two lane road, on-coming traffic may consist of one vehicle passing another. The self vehicle must predict whether the on-coming vehicle in the self vehicle lane will return to its own side before a head-on collision occurs. On a one lane road, it may be necessary for the self vehicle to pull over and let an on-coming vehicle pass, or wait for the on-coming vehicle to pull over so that the self vehicle can pass. On a narrow mountain road, it may be necessary to back up to a place where it is wide enough for two vehicles to pass each other.

Learn from experience and from human instructors

Adjust behavior to situation and priorities. Use reward and punishment from human instructors to learn skills and behaviors. Use experience from multiple simulated scenarios to learn from experience.

6. METRICS AND MEASURES

A metric is a unit of measure. Examples include the meter, the second, the kilogram, the volt, Plank's constant, and Avogadro's number.

Measurements are made by comparing something against the unit of measure. A measurement can be made of the length of the coastline of the British Isles, the height of the Eiffel Tower, the mass of the Queen Mary, the length of a day, or the charge on an electron. There are many parameters related to measurement including accuracy, precision, resolution, observability, and uncertainty.

What is it about intelligent systems that can be measured? If an intelligent system is defined as a system with the ability to act appropriately in an uncertain environment, then we can measure the appropriateness of its behavior. And, if appropriate behavior is defined as that which increases the likelihood of achieving a goal, then the ability of a system to achieve goals in an uncertain environment is a measure of intelligence.

At least three things are required to measure the ability of a system to achieve goals. First, we need to define the goals and set criteria for achieving them. Second, we need to provide an environment in which to make the measurements. Third, we need to define a procedure for scoring performance that takes into account the difficulty of the goals, and the complexity and uncertainty of the environment

What kinds of measurements can be used to measure performance? One possibility is to develop one or more benchmark tests, and measure speed, accuracy, efficiency, level of difficulty, and cost. These measurements can then be weighted for importance and summed to provide an overall score.

Another approach is to devise competitions wherein different intelligent systems can compete against each other for a score. Competitions can involve direct physical interactions such as in football or tennis, measurements of time as in skiing or bobsleding, or competitions that consider both style and difficulty as in ice skating, diving, and gymnastics. Again, performance measurements can be weighted for importance and summed to provide a score.

What kind of metric can be used to measure the performance of an intelligent systems? One possible metric is the performance of a human being. Another possible metric is the performance of a standard baseline system. In either case, the performance of the intelligent system under test can be compared with the performance of a human being (or baseline system) under similar conditions. The difference in performance, the level of difficulty of the test, and the weighting for importance of the test all combine to give a score.

Measures of performance can be devised for subsystem performance, individual system performance, or group or team performance. For example, for subsystems, benchmark tests can be devised to measure the performance of sensory processing algorithms, world model predictors, or behavior generation planners. One might measure the difference between predictions and observations, or the difference between plans and actions. Benchmark tests can also be devised to measure the accuracy of knowledge about the world. For example, one can measure the difference between perceived terrain geometry derived from sensors and ground truth from calibrated test courses. One can measure the latency between requesting and receiving information about the world. Individual system performance can be measured and scored against standard tasks that are typically required of human scout vehicles. Similarly, team performance can be measured and scored in war games wherein opposing forces are tested in battle fighting scenarios.

What is needed?

Calibrated test facilities are needed to test the performance of sensors and systems in the field under realistic conditions. High fidelity simulation facilities are needed to generate repeatable test data for software debugging and testing. Data from calibrated sensors, mixed with a known noise, and accompanied by ground truth are needed to test sensory processing and world modeling algorithms. World model data with values assigned to entities and events is needed to test behavior generation planning and control algorithms.

Large scale test and training facilities are needed to test performance of systems in large scale operations and to develop tactics and training for integration of autonomous systems with manned forces. A wide variety of benchmark tests and competitions are needed to test intelligent system performance under a wide variety of environmental conditions. A rigorous regimen of testing, debugging, and reliability engineering will be needed before intelligent systems become robust enough to operate reliably under a wide variety of operational conditions.

6. REFERENCES

[1] Albus, J.S., “Outline for a Theory of Intelligence,” IEEE Transactions on Systems, Man and Cybernetics, Vol. 21, No. 3, pgs. 473-509, May/June, 1991

[2] Albus, J. *4D/RCS: A Reference Model Architecture for Demo III, Version 1.0, NISTIR 5994,* Gaithersburg, MD, 1998

[3] Shoemaker, C., Bornstein, J., Myers, S., and Brendle, B. “Demo III: Department of Defense testbed for unmanned ground mobility,” *SPIE Conference on Unmanned Ground Vehicle Technology, SPIE Vol. 3693*, Orlando, FA, April, 1999

A Standard Test Course for Urban Search and Rescue Robots

Adam Jacoff, Elena Messina, John Evans
Intelligent Systems Division
National Institute of Standards and Technology
Gaithersburg, MD 20899-8230

ABSTRACT

One approach to measuring the performance of intelligent systems is to develop standardized or reproducible tests. These tests may be in a simulated environment or in a physical test course. The National Institute of Standards and Technology is developing a test course for evaluating the performance of mobile autonomous robots operating in an urban search and rescue mission. The test course is designed to simulate a collapsed building structure at various levels of fidelity. The course will be used in robotic competitions, such as the American Association for Artifical Intelligence (AAAI) Mobile Robot Competition and the RoboCup Rescue. Designed to be highly reconfigurable and to accommodate a variety of sensing and navigation capabilities, this course may serve as a prototype for further development of performance testing environments. The design of the test course brings to light several challenges in evaluating performance of intelligent systems, such as the distinction between "mind" and "body" and the accommodation of high-level interactions between the robot and humans. We discuss the design criteria for the test course and the evaluation methods that are being planned.

KEYWORDS: *performance metrics, autonomous robots, mobile robots, urban search and rescue*

1. INTRODUCTION

The Intelligent Systems Division of the National Institute of Standards and Technology is researching how to measure the performance of intelligent systems. One approach being investigated is the use of test courses for evaluating autonomous mobile robots operating in an urban search and rescue scenario. Urban search and rescue is an excellent candidate for deploying robots, since it is an extremely hazardous task. Urban Search and Rescue (USAR) refers to rescue activities in collapsed building or man-made structures after a catastrophic event, such as an earthquake or a bombing. Japan has an initiative, based on the RoboCup robots, that focuses on multi-agent approaches to the simulation and management of major urban disasters [1]. The real-world utility and manifold complexities inherent in this domain make it attractive as a "challenge" problem for the mobile autonomous robots community. For a description of the issues pertaining to intelligent robots for search and rescue, see [2].

Figures 1 and 2 illustrate the type of environment that a rescuer has to confront with a collapsed building. There is totally unstructured rubble, which may be unstable and contain many hazards. Victims' locations and conditions must be established quickly. Every passing minute reduces the chances of saving a victim.

This type of environment stresses the mobility, sensing, and planning capabilities of autonomous systems. The robots must be able to crawl over rubble, through very narrow openings, climb stairs or ramps, and be aware of the possibility of collapses of building sections. The sensors are confronted with a dense, variable, and very rich set of inputs. The robot has to ascertain how best to navigate through the area, avoiding hazards, such as unstable piles of rubble or holes, yet maximizing the coverage. The robot also has to be able to detect victims and ideally, determine their condition and location. The robot has to make careful decisions, planning its path and strategy, and taking into account the time constraints.

A near-term measure of success for robots in a search and rescue mission would be to scout a structure, map its significant openings, obstacles, and hazards, and locate victims. The robots would communicate with victims, leaving them with an emergency kit that contains a radio, water, and other supplies, and transmit a map, including victim locations and conditions, to human supervisors. Humans would then plan the best means of rescuing the victims, given the augmented situational awareness.

Search and rescue missions are not amenable to teleoperation due to the fact that most of the radio frequencies are reserved by emergency management agencies. Obstructions and occlusions also diminish the effectiveness of radio transmissions. Tethers are not typically practical in the cluttered environment in which these robots must operate.

2. URBAN SEARCH AND RESCUE AS A ROBOTIC CHALLENGE

A search and rescue mission is extremely challenging and dangerous for human experts. This is a highly unstructured and dynamic environment, where the mission is time critical. Very little *a priori* information about the environment or building may exist. If any exists, it will almost certainly be obsolete, due to the collapse.

Urban Search and Rescue is therefore attractive as a mission framework in which to measure intelligence of autonomous robots. The high degree of variability and unpredictability demand high adaptation and sophisticated decision-making skills from the robots. Robots will need to quickly and continually assess the situation, both in terms of their own mobility and of the likelihood of locating more victims. USAR missions are amenable to cooperation, which can be considered another higher-level manifestation of intelligence. We propose that any robot or team of robots that is able to successfully and efficiently carry out USAR missions would be considered intelligent by most standards.

In the following sections, we will briefly discuss how USAR missions tax specific components of an intelligent system.

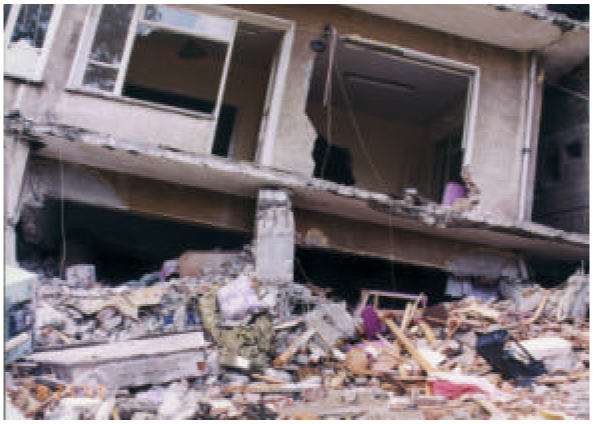


Figure 1: Partially Collapsed Building from Turkey Earthquake

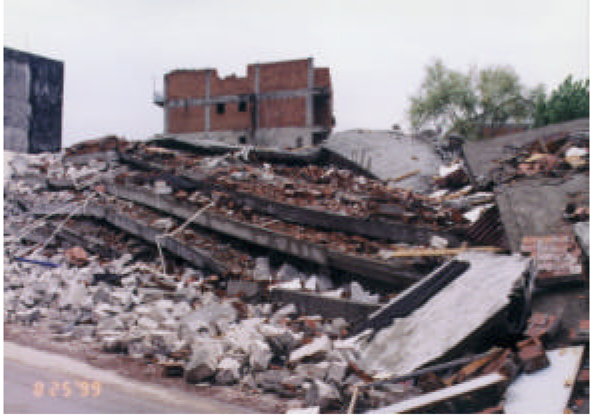


Figure 2: Totally Collapsed Building from Turkey Earthquake

2.1 MOBILITY

As can be seen from Figures 1 and 2, the mobility requirements for search and rescue robots are challenging. They must be able to crawl over piles of rubble, up and down stairs and steep ramps, through extremely small openings, and take advantage of pipes, tubes, and other unconventional routes. The surfaces that they must traverse may be composed of a variety of materials, including carpeting, concrete blocks, wood, and other construction material. The surfaces may also be highly unstable. The robot may destabilize the area if it is too heavy or if it bumps some of the rubble. There may be gaps, holes, sharp drop-offs, and discontinuities in the surfaces that the robot traverses.

2.2 SENSING

In order to be able to explore an USAR site and successfully navigate in this environment, the robot's sensing and perception must be highly sophisticated. Lighting will be variable and may be altogether missing. Surface geometry and materials may absorb emitted signals, such as acoustic, or they may reflect them. For truly robust perception, the robots should emulate human levels of vision.

The presence of victims may be manifested through a variety of signals. The stimuli that the robots have to be prepared to process include

- Acoustic – victims may be calling out, moaning softly, knocking on walls, or otherwise generating sounds. There will be other noises in the environment due to shifting materials or coming from other USAR entities.
- Thermal – a body will emit a thermal signature. There may be other sources of heat, such as radiators or hot water.
- Visual – a multiplicity of visual recognition capabilities, based on geometric, color, textural, and motion characteristics, will be exercised. Recognizing human characteristics, such as limbs, color of skin, clothing is important. Motion of humans, such as waving, must be detected. Confusing visual cues may come from wallpaper, upholstery or curtain material, strewn clothing, and moving objects, such as curtains blown by a breeze.

2.3 KNOWLEDGE REPRESENTATION

In order to support the sophisticated planning and decision-making that is required, the robot must be able to leverage a rich knowledge base. This entails both *a priori* expertise or knowledge, such as how to characterize the traversability of a particular area, as well as gained information, such as a map that is built up as it explores. It must develop rich three-dimensional spatial maps that contain areas it or other robots have and haven't yet seen, victim and hazard locations, and potential quick exit routes. The maps from several robots may need to be shared and merged.

A variety of types of knowledge will be required in order to successfully accomplish search and rescue tasks. Higher-level knowledge, which may be symbolic, includes representations of what a "victim" is. This is a multi-facetted definition, which includes the many manifestations that imply a victim's presence.

2.4 PLANNING

An individual robot must be able to plan how to best cover the areas it has been assigned. The time-critical nature of its work must be taken into account in its planning. It may need to trade off between delving deeper into a structure to find more victims and finding a shortcut back to its human supervisors to report on the victims it has already found.

2.5 AUTONOMY

As mentioned above, it is not currently practical to assume that the robots will be in constant communication with human supervisors. Therefore, the robots must be able to operate autonomously, making and updating their plans independently. In some circumstances, there may be limited-bandwidth communications available. In this case, the robots may be able to operate under a mixed-initiative mode, where they have high-level interactions with humans. The communications should be akin to those that a human search and rescue worker may have with his or her supervisor. It definitely would not be of a teleoperative nature.

2.6 COLLABORATION

Search and rescue missions seem ideally suited for deploying multiple robots in order to maximize coverage. An initial strategy for splitting up the area amongst the robots may be devised. Once they start executing this plan, they will revise and adapt their trajectories based on the conditions that they encounter. Information sharing between the robots can improve their efficiency. For example, if a robot detects that a particular passageway that others may need to use is blocked, it would communicate that to its peers. The robots should therefore collaborate and cooperate as they jointly perform the mission. They may be centrally or decentrally controlled. The robots themselves may all have the same capability, or they may be heterogeneous, meaning that they have different characteristics. Heterogeneous robot teams may apply the marsupial approach, where a larger robot transports smaller ones to their work areas and performs a supervisory function.

3. MEASURING THE PERFORMANCE OF USAR ROBOTS

We have described briefly the requirements for autonomous urban search and rescue robots. We will now discuss approaches to testing their capabilities in achieving a USAR mission.

The approach being taken by the upcoming USAR robot competitions that will use the NIST test course is based on a point system. The goal of the robots is to maximize the number of victims and hazards located, while minimizing the amount of time to do so and the disruption of the test course.

Specifically, the AAAI Mobile Robot competition [1] will use Olympic-style scoring. Each judge will have a certain number of points that can be awarded based on their measuring certain quantitative and qualitative metrics. Robots receive points for

- Number of victims located
- Number of hazards detected
- Mapping of victim and hazard locations
- Staying within time limits
- Dropping off a package to victims representing first aid, a radio, or food and water
- Quality of communications with humans
- Tolerance of communications dropout

They lose points for

- Causing damage to the environment, victims, or themselves (e.g., destabilizing a structure)
- Failing to exit within time limits

In certain sections of the test course, robots are allowed to have high-level communications with humans. These communications must be made visible to the judges. Metrics for evaluating the quality of the communications include "commands" per minute and/or bandwidth used. Fewer commands per minute and less bandwidth per minute receive better scores. Tolerance of communications disruption is an important capability and will be given greater difficulty weighting. A team may request that the judges simulate communication disruptions at any point in order that the robots demonstrate how to recover. Examples of recovery would be to move to a location where there is better chance of communication, making decisions autonomously instead of consulting humans, or utilizing companion robots to relay the information to the humans.

For teams consisting of multiple robots, the advantage of cooperating or interacting robot must be demonstrated. This can be either in performing the task better, or performing the task more economically. Multi-robot teams should have a time speedup that is greater than linear, or may be able to perform the tasks with less overall power consumption or cost. The scoring will factor in the number of robots, types of robots, types or mixture of sensors, etc., in determining the performance of a team.

The RoboCup Rescue competition, sponsored by Robot World Cup Initiative, takes an evaluation benchmarking approach. Initially, there are 3 benchmark tasks. The current tasks are victim search, victim rescue, and a combination of victim search and rescue. Additional ones will be added as the competition and participants evolve. The RoboCup Rescue includes a simulation infrastructure in which teams can compete, as well as the use of the NIST test course.

Their evaluation metrics are still under development. Examples of criteria that have been published on their web site [4] include:

- Recovery rate, expressed as percentage of victims identified versus number under the debris.
- Accuracy rate, computed as the number of correctly identified victims divided by the total number of identified victims.
- Operational loading, which is the number of operations that a human has to perform in order to enable to robots to perform their tasks.
- If rescuing victims, the total time it takes to rescue all victims.
- Total damage caused to victims in attempting to rescue them.

4. THE TEST COURSE DESIGN

The test course which NIST designed for the AAAI Mobile Robot Competition was designed with three distinct areas of increasing verisimilitude and difficulty. Overall, the course is meant to represent several of the sensing, navigation, and mapping challenges that exist in a real USAR situation. As discussed above, these are challenges that correlate well with general characteristics desirable in mobile, autonomous robots that may operate in other types of missions. In the design of the course, tradeoffs were made between realism and reproducible and controlled conditions. In order to be able to evaluate the performance of robots in specific skill areas, certain portions of the course may look unrealistic or too simplified. This idealization is necessary in order to abstract the essential elements being exercised, such as a the ability to deal with a particular sensing challenge.

Given the controlled conditions that the test course provides, it is possible to have multiple robots or teams face the identical course and have their performances compared. This should yield valuable information about what approaches to robotic sensing, planning, and world modeling work best under certain circumstances.

The course is highly modular, allowing for reconfiguration before and during a competition. Judges may swap wall panels that are highly reflective for some that are fabric-covered, for example, or victims may be relocated. This reconfigurability can serve to avoid having robot teams "game" the course, i.e., program their robots to have capabilities tailored to the course they've seen previously. The reconfigurability can serve to provide more realism as well. A route that the robot used previously may become blocked, forcing the robot to have to find an alternative way.

The three areas of the course are described below. Note that the use of color in the names of the section is for labeling purposes only and does not mean that the courses are primarily colored in their namesake color. A representative schematic of the test course is shown in Figure 3, at the end of this paper.

4.1 YELLOW COURSE

Given the fact that participating teams, at least initially, will primarily be from universities that may not have access to new agile robotic platforms, one design requirement was to have an area within the course where the mobility challenges are minimal. We call this area the "Yellow course." The floor of the yellow course is flat and of uniform material. Passageways are wide enough to permit large robots, up to about 1 meter diameter, to pass easily.

Yet the Yellow course allows teams with sophisticated perception and planning to exercise their robots' capabilities. Some sensing challenges are as difficult in this section as in the others. There will be highly reflective and highly absorbent material on walls. Certain wall panels will be clear Plexiglas, whereas others will be covered in brightly patterned wallpaper. Some areas may be dimly lit or accessible only from one direction. Victims will be represented in all modalities (i.e., acoustically, visually, through motion, thermally, etc.) and may be hidden from view under furnishings or in closed areas.

4.2 ORANGE COURSE

The Orange course is of intermediate difficulty. A second story is introduced, and there are routes that only smaller robots may pass through. The robots may have to climb stairs or a ramp in order to reach victims. Flooring materials of various kinds, such as carpeting, tile, and rubber, are introduced. Hazards, such as holes in the floor, exist. In order to be effective, the robot will have to plan in a three-dimensional space. Larger robots will be able to navigate through some portions of this course, but not all.

4.3 RED COURSE

The Red course poses the most realistic representation of a collapsed structure. We do not anticipate that any of the contestants will be able to successfully complete the red course in the first or perhaps even second years. However, this section provides a performance goal for the teams to strive for. In the Red section, piles of rubble abound, lighting is minimal or non-existent, and passageways are very narrow. The course is highly three-dimensional, from a mapping perspective. Not only are there two floors, but the rubble piles that the robot has to traverse may need to be mapped as well. Passageways under the rubble or through pipes may have to be used by the robots to reach certain areas or to get closer to victims. There are some portions of this course that can be traversed by the larger class of robots, but they would not be able to reach most of the victims. Larger robots would be best suited in marsupial configurations in this area.

5. CONCLUSION

An Urban Search and Rescue application for autonomous mobile robots poses several challenges that can be met only by highly intelligent systems. The variability, risk, and urgency inherent in USAR missions makes this a good framework in which to begin measuring performance in controlled and reproducible situations. We believe that the test course we are developing can serve to elucidate performance measures for overall systems, as well as for components of intelligent systems.

6. REFERENCES

[1] http://www.aic.nrl.navy.mil/~schultz/aaai2000/menu-bar-vert.html

[2] Casper, J., and Murphy, R.R., “Issues in Intelligent Robots for Search and Rescue,” SPIE Ground Vehicle Technology II, Orlando, FL, April 2000.

[3] Kitano, H., et al., ‘RoboCup Rescue: Search and Rescue in Large-Scale Disasters as a Domain for Autonomous Agents Research,” Proceedings of the IEEE Conference on Man, Systems, and Cybernetics, 1999.

[4] http://www.robocup.org/games/36.html

ACKNOWLEDGEMENTS

The development of the USAR test course was carried out with the support from the Defense Advanced Research Agency (DARPA) MobileAutonomous Robot Software Program and the Naval Research Laboratory.

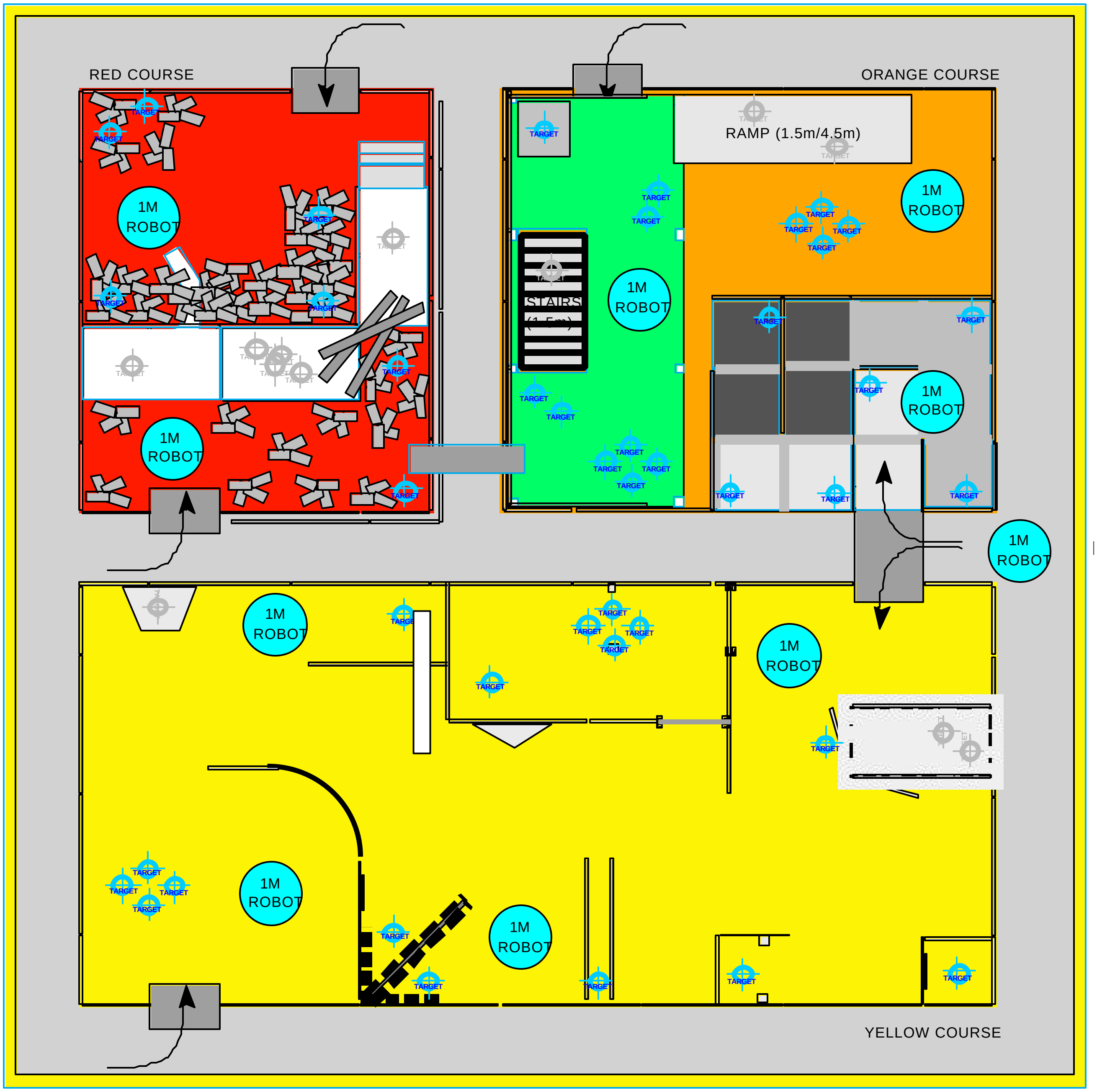


Figure 3: Overall USAR Test Course Layout

Overall dimensions are approximately 17 by 20 meters

1M ROBOT	1 Meter Robot drawn to show scale
⌖	Represents a victim “signature”, such as a thermal emission, clothing, or other manifestation

Assessment of the NIST Standard Test Bed for Urban Search and Rescue

Robin Murphy, Jenn Casper, Mark Micire, Jeff Hyams
Computer Science and Engineering
University of South Florida
Tampa, FL 33620
{murphy, jcasper, mmicire, hyams}@csee.usf.edu

ABSTRACT

The USF team in the 2000 AAAI Mobile Robot Competition had the most extensive experience with the NIST Standard Test Bed for USAR. Based on those experiences, the team reports on the utility of the test bed, and makes over 20 specific recommendations on both scoring competitions and on future improvements to the test bed.

1 INTRODUCTION

A team of three operators and two robots from the University of South Florida (USF) tested the NIST standard test bed for urban search and rescue (USAR) as part of the 2000 AAAI Mobile Robot Competition USAR event. The test bed consisted of three sections, each providing a different level of difficulty in order to accommodate most competitors (see Fig. 1). The easiest section, Yellow, contained mainly hallways, blinds, and openings to search through. The course could be traversed by a Nomad type robot. The intermediate Orange Section provided more challenge with the addition of a second level that was reachable by stairs or ramp. Other challenges included those found in the yellow as well as some added doors. The Red Section was intended to be the most difficult. It contained piles of rubble and dropped floorboards that represented a pancake-like structure. The Orange and Red sections clearly required hardware that was capable of traveling such spaces.

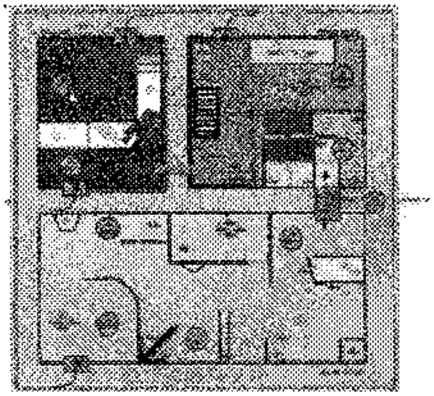


Figure 1: Overview of the NIST USAR arena.

In addition to USF, three other teams entered the AAAI competition's USAR event: Kansas State, Swarthmore College, and University of Arkansas. The Kansas State team dropped out due to hardware failures on site. The Swarthmore and Arkansas teams fielded Nomad scout types of robots that operated only in the Yellow Section. The performance of each team is unclear as the judges did not record how many victims were found and how many victims were missed. At the time of publication of this paper, the awards for the event were under protest. Swarthmore had a single robot which attempted to enter a room, perform a panoramic visual scan for possible victims, mark the location on a map, and then enter another room and so on. At the conclusion of their allotted time, the robot was retrieved and the contents of the map was made available to the judges. They entered one room successfully and it is believed they identified up to two surface victims. The Arkansas team used two Nomad scout type robots; however, each robot was physically placed in a room, and the team was allowed to repeatedly move and reset the robots as needed. The Arkansas team found at least one victim, and communicated this by repeatedly ramming the mannequin.

The USF team used two outdoor robots: 1) a RWI ATRV with sonar, video, and a miniature uncooled FLIR and 2) a customized RWI Urban with a black and white camera, color camera, and sonars. This was intended to be a marsupial pair, but the transport mechanism for the team was still under construction at the time of the competition. The USF team used a *mixed-initiative* or *adjustable autonomy* approach: each platform was teleoperated for purposes of navigation but ran a series of software vision agents for autonomous victim detection: motion, skin color, distinctive color, and thermal region. The user interface displayed the extraction results from each applicable agent and highlighted in color whenever the agent found a candidate. A fifth software agent ran on the ATRV which fused the output of the four agents, compensating for the physical separation between the video and FLIR cameras. It beeped the operator when it had sufficient confidence in the presence of a victim, but the beeping had to be turned off due to a high number of false positives generated by the audience. The ATRV found an average of 3.75 victims per each of the four runs recorded, while the Urban found an average of 4.67 victims. A fifth run was not recorded and no data is available.

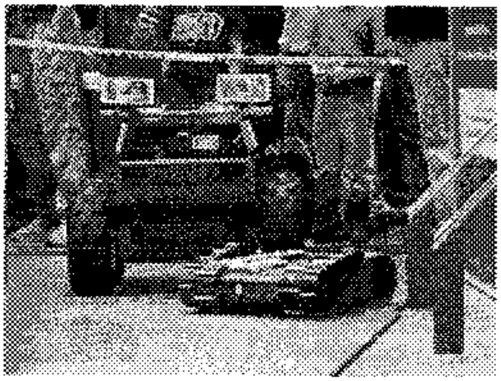


Figure 2: The USF USAR robot team, Fontana (ATRV) and Klink (Urban) (named after two women Star Trek writers).

In addition to participating in the competition (both a preliminary and a final round), the USF team hosted three complete exhibition runs as part of the AAAI Robot Exhibition Program and did numerous other partial exhibitions for the news media at the request of AAAI. The other teams did not exhibit. As such, the USF team had the most experience with the most difficult sections of the test bed and can claim to represent user expertise.

This paper discusses the NIST test bed from the USF experience, and makes recommendations on scoring, improving the test bed, and staging a more USAR-relevant event at RoboCup Rescue in 2001.

2 ASSESSMENT OF THE THREE SECTIONS

The NIST test bed is an excellent step between a research laboratory and the rigors of the field. For example, USF has a USAR test bed (Fig. 3), but it is somewhere between the Yellow and Orange sections in difficulty and cannot provide the large scale of the NIST test bed. One advantage is that the test bed sections can be made harder as needed. An important contribution that should not be overlooked is that the test bed appeared to motivate researchers we talked to: it was neither too hard nor too trivial. This is quite an accomplishment in itself.

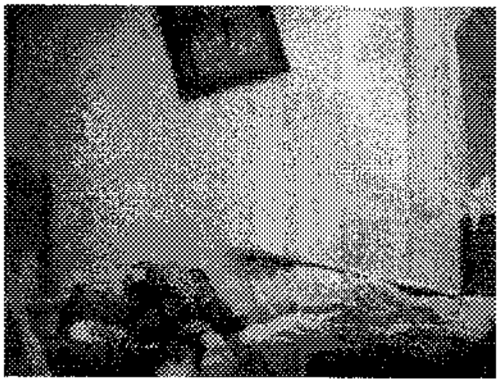


Figure 3: The USF USAR testbed, a mock-up of a destroyed bedroom.

2.1 *Yellow*

The USF team did not compete or exhibit in the Yellow Section, entering only for about 1 hour of practicing collaborative teleoperation. Our assessment was that the section was far too much of an office navigation domain- the over-turned chair in one of the rooms was the only real surprise. Only one room had a door and neither Swarthmore nor Arkansas reached it. The arena was at about the level of complexity seen in the Office Navigation Event thread of the AAAI Robot Competition in the mid-1990's.

2.2 *Orange*

The Orange Section consisted of a maze plus a second story connected by a ramp and stairs. Unlike the Yellow Section, the doorways into the Orange and Red Sections had cross-members crowning the doorway at about 4 feet high. This added some feel of confined space. The USF robots entered a very confined maze of corridors to find a surface victim. The Urban served as point man, exploring first, then guiding the ATRV if it found something requiring confirmation or IR sensing. The maze had hanging Venetian blinds in the passage way, and the Urban almost got the cord tangled in her flipper.

The Orange Section also had two forms of entry in the main search area after the robots had navigated the maze. One entry was through the X made by cross-bracing the second story. The Urban could navigate under the cross-bracing, but the ATRV could not. The second form of entry was through a door on hinges. The Urban pushed the door open for the ATRV to enter the main search space (Fig. 4). The Urban attempted to climb the stairs, but the first step was too high for the design. (A Matilda style robot also attempted to climb the stairs but could not either.) It went to the ramp and climbed to the second story.

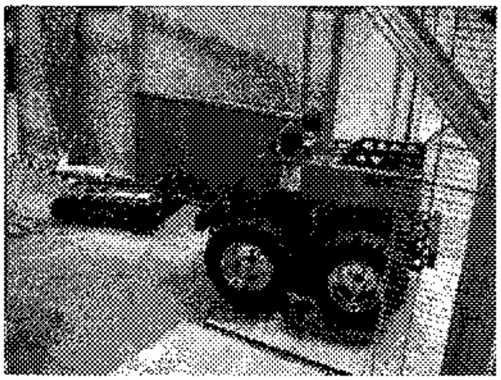


Figure 4: The Urban holds the door for the ATRV in the Orange Section.

The USF robot was able to avoid negative obstacles (a stairwell and uncovered HVAC ducting in the floor of the second story) to find victims on the second story (Fig. 5). The modified Urban actually flipped its upper camera onto the HVAC hole and peered inside the duct. This shows the utility of having multiple sensors and in different locations.

The Orange Section is also to be commended for providing some attributes of 3D or volumetric search. For example, an arm was dangling down from the second story and should have been visible from the first floor. Note that the dangling arm posed a classic challenge between navigation and mission. The mission motivates the robot or rescuer to attempt to get closer and triage the victim, while the navigational layout prevents the rescuer from approaching without significantly altering course, and even backtracking to find a path.

2.3 *Red*

The Red Section at first appeared harder (Fig. 6), however, in practice it was easier for the ATRV than the Orange Section due to more open space. The floor was made up of tarps and rocks on plywood. The ATRV and Urbans were built for such terrain. The Red Section contained two layers of pancaking, with significant rubble, chicken wire, pallets, and pipes creating navigation hazards for the Urban. Only about 30% of the area was not accessible to the larger ATRV due to the large open space.

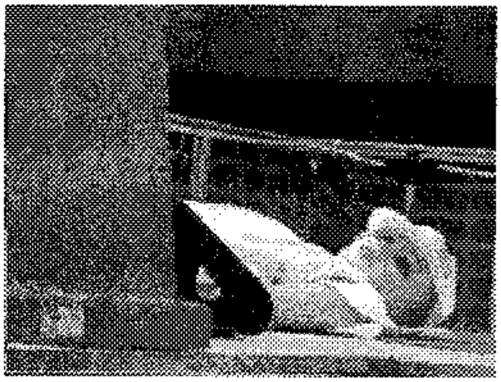


Figure 5: Close up of victim lying on the second floor of the Orange Section.

One nice attribute of the Red Section is that it lends itself to booby-traps. The pancake layers were easily modified between runs to create a secondary collapse when the Urban climbed to the top. Using current technology, the Urban operator and/or software agents could not see any signs that the structure was unstable.

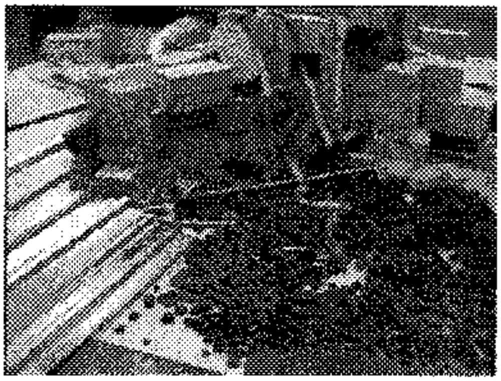


Figure 6: Overview of the Red Section.

3 RECOMMENDATIONS ON SCORING

The AAAI Competition did not use any metric scores for their USAR event, relying entirely on a panel of four judges, none of whom had any USAR experience. The AAAI Competition had published metrics prior to the competition that were to be used in scoring,[5] but did not use those metrics on site and the scoring was subjective. The published metrics appeared to be a good first start (with our reservations given below) and no reason was given why AAAI abandoned them.

1. Use quantitative scoring, at least as a basis for the com-

petition. The scores might be modified by a qualitative assessment of the AI involved, but there should be a significant numerical aspect to the scoring.

2. Distribute victims in same proportions as FEMA statistics given in FEMA publication USFA/NFA-RS1-SM1993 and award points accordingly. Detecting a surface victim and an entombed victim require much different sensing and intelligence.

Surface	50%
Lightly trapped	30%
Trapped in void spaces	15%
Entombed	5%

3. Have a mechanism for unambiguously confirming that the victims identified were identified. It was not clear to the audience when a victim had been correctly detected or when the robot had reported a false positive. Perhaps an electronic scoreboard showing the number of false positives and false negatives (missed victims) could be displayed and updated during the competition. (Swarthmore used beeping and USF flashed the headlights. The judges appeared to accept that if there was a victim in the general direction of the robot's sensors at the time of the announced detection that a victim had been found. In the case of USF, only one judge took time during the competition look at the technical rescue display workstation, which provided both the sensor data and the fused display, to confirm what the robot was seeing.)

4. Points for the detection of a victim should also depend on the time at which the technical rescue crew is informed of the discovery and the accuracy of the location, either in terms of absolute location or a navigable path for workers to reach the victim. Robots which overcome inevitable communications problems by creating a relay of "comms-bots" or returning to locations where broadcasting worked are to be rewarded. (The Swarthmore robot beeped when it thought it found a victim, but in terms of truly communicating that information to rescue workers, it stored the location of all suspected victims until the competition was ended. In practice, if the robot had been damaged, the data would have been lost. Also, the map was not compared quantitatively to the ground truth.)

5. Contact with the victims should be prohibited unless the robot is carrying a biometric device that requires contact with the victim. In that case, the robot should penalized or eliminated from competition if contact is too hard or otherwise uncontrolled. (The Arkansas robots repeatedly struck the surface victim it had detected.)

6. Fewer points should be awarded for finding a disconnected body part (and identifying it as such) than for finding a survivor.

7. Require the robots to exit from the same entry void that they used for entry. This is a strict requirement for human rescue workers in the US, intended to facilitate accounting for all resources. (The AAAI Competition permitted exiting from anywhere on the grounds that the robot may need to find a clear spot to communicate its results.)

8. Have all competitors start in the same place in the warm zone, and do not permit them to be carried by human operators inside the hot zone. The exception is if the robot has to be carried and inserted in an above grade void from the outside. (Swarthmore and Arkansas manually placed their robots in the yellow section, with Arkansas actually placing their robots within specific rooms in the yellow section.)

9. Do not permit human operators to enter the hot zone and reset or move robots during the competition. (Arkansas team members repeatedly entered the hot zone to reboot errant robots and to physically move robots to new rooms to explore.)

10. Have multiple runs, perhaps a best of three rounds approach used by AUVSI. (NIST "booby-trapped" the Red Section after the AAAI Preliminary Round, making it extremely easy to create a secondary collapse. This was done to illustrate the dangers and difficulties of USAR. However, if the AAAI rules had been followed, this would have resulted in a significant deduction of points from the USF team, and quite a different score between runs. The difficulty of the courses should be fixed for the competition events, and changed perhaps only for any exhibitions.)

It should be clear from the above recommendations that a quantitative scoring system which truly provides a "level playing field" is going to be hard to construct. Unlike RoboCup, where the domain is a game with accepted rules and scoring mechanisms, USAR is more open. In order to facilitate the relevance of the competition to the USAR community, we recommend that scoring mechanisms be derived in conjunction with USAR professionals outside of the robotics community and with roboticists who are trained in USAR. We propose that a rules committee for RoboCup Rescue physical agent be established and include at least one representative from NIST, NIUSR, and one member of the research community who had worked and published in USAR.

4 RECOMMENDATIONS FOR IMPROVING THE NIST TESTBED

The NIST testbed was intended to be an intermediate step between a research laboratory and a real collapsed building. The three sections appeared to be partitioned based on navigability, rather than as representative cases of severity of building collapses or perceptual challenges. For example, the basic motivation for the Yellow versus the Orange and Red Sections appeared to be to engage researchers with traditional indoor robot platforms (e.g., Nomads, RWI B series, Pioneers, and so on). An alternative strategy might be to consider each section more realistically, where the Yellow Section would be a structurally unsound, but largely navigable, apartment building, the Orange Section might be an office building in mixed mode collapse such as many of the buildings in the 1995 Hanshin-Awaji earthquake, and the Red Section might be a pancake collapse such as seen in the front of the Murrow building at the Oklahoma City bombing. This approach would permit traditional indoor robot platforms to navigate, but require advances in detection of unfriendly terrain such as throw rugs or carpet, doors, etc.

4.1 *For All Sections*

In addition to the suggestions made above, we offer some possible improvements to the test bed:

1. Create void spaces in each section more typical of USAR (Fig. 7). In particular, there were no lean-to and V void spaces in any of the 3 sections. The red section did have some light pancaking. Victims in even the Yellow Section should be placed behind furniture and occluded by fallen furniture or even sheet-rock or portions of the ceiling.

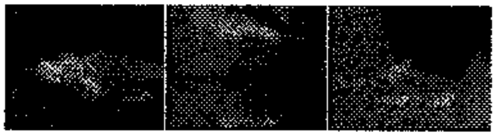


Figure 7: Infrared images of a lightly trapped, void trapped, and entombed victim.

2. Put tarps and high powered lights ("beams of sunlight") over portions of all courses to create significant changes in lighting conditions, most especially darkness. As it stands now, the testbed is a poor test of the utility of infra-red.
3. Entries were all doors at grade. Many voids are actually above grade, irregular, and have been knocked in the wall, even in buildings that have not collapsed. Each section should have one or more above grade entry voids from the "outside". This will support the testing of concepts for automating the reconnaissance and deciding how to deploy resources, as per the rescue and recovery of lightly trapped victims, use of reconnaissance results to locate lightly trapped victims, and searching void spaces after hazard removal phases of a structural collapse rescue.[4]
4. Each section should contain more human effects. For example, the Yellow and Orange Sections should have throw rugs on the floors, fallen debris such as magazines, books, bills, toys, etc. Otherwise, the Yellow Section is actually easier than the Office Navigation thread in the AAAI competitions during the mid-1990's.
5. Each section should contain real doors with door knobs or at least the commercial code handles for disabled access. The doors in the Yellow and Orange section were both easily opened panels. (USF was able to easily identify the swinging door in the Orange Section and use the Urban to open the door for the ATRV to pass through. None of the other teams got to the room with the door in the Yellow Section). All rooms in any section should have doors and some of those doors should be off their hinges or locked. This will test the advances in object recognition, reasoning, and manipulation.
6. If possible, victims should produce a more realistic heat profile than a heating pad. This is needed for detection and to test advances in assessment of the context of the victim (how much they are covered, etc.).

4.2 *For the Orange and Red Sections*

1. Cover everything with dust to simulate the cinder block and sheet-rock dust that commonly covers everything in a building collapse. Victims who are alive often move enough to inadvertently shake off some of this dust, making color detection a very important component of victim detection. (USF used a "distinctive color detector" as one of their four vision agents. The distinctive color agent looked for regions of color that were different than the average value. This appeared to work during the competition for the Red Section, which was less colorful (no wallpaper, etc.), but there wasn't enough data to draw any statistical conclusions.)
2. Make the surfaces uneven. All the surfaces were level

in their major axis; even the ramp in the Orange Section was flat, not canted to one side.

3. Use real cinder blocks. The USF Urban was able to move the faux cinder blocks on the ramp in the Orange Section rather than navigate around (Fig. 8).

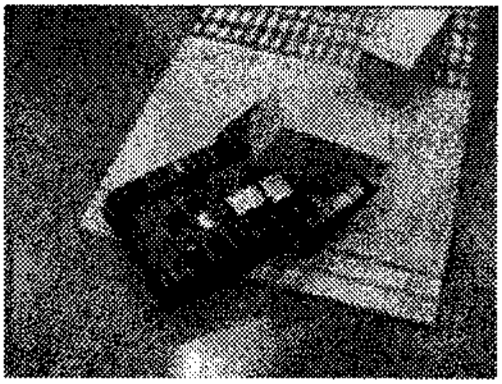


Figure 8: The Urban has pushed the cinder block around rather than traversed over it.

4. Make a "box maze" for entry to introduce more confined space. Rescue workers who are certified for confined space rescue use a series of plywood boxes which can be connected together to form long, dark, confined mazes. The mazes are easily reconfigured. A similar box maze could be constructed from the lightweight paneling material.

5. The terrain of both sections was still fairly easy compared to the field, and dry. Perhaps as robot platforms evolve, the courses should contain water.

5 OTHER SUGGESTIONS

The testbed is primarily intended to be a standard course for experimentation. The AAAI Competition did not especially further experimentation, as that the competition judges collected no metric data. However, the AAAI Competition performed a valuable service by illustrating the potential conflict between science and exhibitions. The public viewing interfered with testing and validating aspects of AI in two different ways. Public viewing may also lead to a tendency towards "cuteness" at the expense of showing direct relevance to the USAR community.

5.1 *Viewing versus Validation*

The conflict between spectator viewing and validation is best seen by the following example. One of the USF vision agents identified large regions of heat using a FLIR, then fused that data with regions of motion, skin color, and distinctive color extracted by software agents operating on video data. If there was a sufficiently strong correlation, the operator interface began beeping to draw the operator's attention to the possibility of a survivor. (The RWI supplied user interface for the Urban requires almost full attention just to navigate, detection assistance is a practical necessity.)

Unfortunately, the test bed has Plexiglas panels to facilitate judge and spectator viewing. AAAI permitted spectators to ring the sections during the competition. Between the low height of walls and the Plexiglas, these spectators were visible and produced color, motion, and IR signatures even when the USF robots were facing interior walls due to views of exterior walls in other sections. As a result, USF had to turn off automatic victim notification through audio and rely strictly on color highlighting in the display windows.

A long-term solution is to insert cameras into the testbed to record, map, and time robot activity as well as broadcast the event to a remote audience. The competition chair stated that the audience should be allowed viewing access on the grounds that rescue workers would be visible in a real site. We note that at a "real site", access to the hot zone is strictly controlled and very few, certified technical rescue workers are permitted in the hot and warm zones. The rest must wait in the cold zone at least 250 feet from the hot zone.[4] Also, at a real site, walls would have blocked views of people versus the half height panels.

Second, in order to record and broadcast the event, photographers and cameramen were permitted in the ring during the exhibitions and competition. During the exhibition, a cameraman repeatedly refused to move out of the robots' way. When the robot continued on, it almost collided with the video recorder.

Therefore, we recommend:

1. At least the Red Section should be fitted with walls and ceilings to block the view of non-testbed elements and the audience.

2. The test bed sections should be fitted with cameras and no one should be permitted in the test bed during timed events. If a robot dies (such as the USF Urban due to a faulty power supply or the Arkansas robots due to software failures), the robot should remain there until the session is complete.

5.2 *Relevance to the USAR Community*

In our opinion, the AAAI Competition missed several opportunities to show a clear relevance of the NIST test bed and robots to the USAR community. As discussed earlier, USAR professionals should be involved in setting the rules as well providing realistic scenarios. In general, any further competition venues, such as RoboCup Rescue, should actively discourage anything that might be construed as trivializing the domain. For example, Swarthmore costumed their robot as a Florence Nightingale style nurse, which rescue workers were likely to find offensive. Likewise, a handwritten "save me" sign was placed next to a surface victim.

The test bed may also miss relevance to the USAR field if it focuses only on benchmarking fully autonomous systems rather than on more practicable mixed-initiative (adjustable autonomy) systems. The Urban type of robot in a hardened form capable of operating in collapsed structures must be controlled off-board: they do not have sufficient on-board disk space to store vision and control routines. Therefore, the test bed should measure communications bandwidth, rate, and content in order to categorize the extent of a system's dependency on communications. Also, the test bed should include localized communications disrupters to simulate the effect of building rubble on communications systems.

6 CONCLUSIONS

Based on our five complete runs in the NIST test bed at AAAI and numerous informal publicity demonstrations, the USF team has had the most time running robots in the test bed. We conclude that the NIST test bed is an excellent halfway point between the laboratory and the real world. The test bed can be evolved to increasingly difficult situations. The initial design appears to have focused on providing navigational challenges, and it is hoped that future versions will add perceptual challenges.

Our recommendations fall into four categories. First, scoring or validation will be a critical aspect of the test bed. The AAAI competition did not implement a quantitative scoring system and thus provides no feedback on what are reasonable metrics. We recommend many metrics, but our guiding suggestion is to get knowledgeable representatives from the USAR community involved in setting up scenarios and metrics. In particular, we note that the victims should be distributed in accordance to FEMA statistics for surface, lightly trapped, void trapped, and entombed victims, and then points awarded accordingly. One major issue that arose from the USF team trying to reconstruct its rate of victim detection was that there needs to be an unambiguous method for signaling when a victim has been detected. Another aspect of scoring is to complement the proposed AAAI "black box" (external performance) metrics with a rigorous "white box"(software design and implementation) evaluation. Second, the test bed should be made more representative of collapsed buildings. We believe this can be done without sacrificing the motivation for the different sections. For example, all sections need to have void spaces representative of the three types discussed in the FEMA literature (lean-to, V, and pancake). The Yellow Section can still have a level, smooth ground plane but the perceptual challenges can be more realistic. Third, the test bed should resolve the inherent conflict between spectator viewing and validation. We believe this can be done by inserting cameras into the test sections as well as adding tarps and walls. Finally, we strongly urge the mobile robotics community to concentrate on making the NIST test bed and any competition venue which uses the test bed to be relevant to the USAR community. The community should resist the tendency to "be cute" and instead use the test bed as a means of rating mixed-initiative or adjustable autonomy systems that can be transferred to the field in the near future as well as the utility of fully autonomous systems.

ACKNOWLEDGMENTS

Portions of the work reported in this paper were conducted as part of work under the DARPA BAA 9835 program and through a donation from SAIC. The authors would like to thank AAAI, the USF Department of Computer Science and Engineering, and the USF Engineering College for travel support, and Mark Powell and Brian Minten for their help in programming and maintaining the robots.

References

[1] *Rescue Systems I*. National Fire Academy, 1993.

[2] *Technical Rescue Program Development Manual*. United States Fire Administration, 1996.

[3] *Standard on Operations and Training for Technical Rescue Incidents*. National Fire Protection Association, 1999.

[4] Casper, J., Micire, M., Murphy, R.R. "Issues in Intelligent Robots for Search and Rescue," in SPIE Ground Vehicle Technology II, 2000.

[5] http://www.cs.swarthmore.edu/ meeden/aaai00/ contest.html#usar

PERFORMANCE METRICS IN THE AAAI MOBILE ROBOT COMPETITIONS

Alan C. Schultz
Naval Research Laboratory
Code 5515
Washington DC 20375
schultz@aic.nrl.navy.mil

Abstract

In this paper, performance metrics used in the AAAI Mobile Robot Competition and Exhibition over the nine years of the contest are compared. Performance metrics have tended from more explicit quantitative measures to more qualitative measures. The author believes that this trend is the result of more complex tasks where more aspects need to be measured. The paper will end by claiming that competitions that are to measure intelligence in robots should include tasks that require adaptation and learning, which the author believe are the hallmarks of intelligence.

Keywords: mobile robot contests, competitions, metrics, multi-agent robotics, learning and adaptation, autonomous robots.

1. Introduction

Although there are several annual mobile robot competitions, the American Association for Artificial Intelligence's (AAAI) Mobile Robot Competition and Exhibition has distinguished itself by attempting to reward those contestants that show the greatest amount of "intelligence" in solving a given task [1-6].

Since this event is organized as a competition, metrics are required for measuring performance in a task that also try to measure the degree of intelligence the robot has exhibited.

Because the contest is organized under the sponsorship of the AAAI, a goal of the competition is to foster research and education in artificial intelligence. As such, tasks selected for the competition were picked because they required some level of "intelligent" behavior or knowledge representation.

2. Early Years: Quantitative Metrics

In the first two years of the competition, less explicit quantitative metrics were used. However, many teams complained that the rules were not explicit enough leading to ambiguities in scoring and in problems interpreting the rules. Starting in the third and subsequent years of the competitions, more explicit and published quantitative measures of performance in the task have been used. It was assumed that completion of the task itself was indicative of intelligence. Points would be awarded to various activities (subtasks) and for abilities and competencies achieved by the robot. The final score would be a summation on the individual points. In some events, points could be removed for exhibiting some undesired behavior. Depending on the task, time would be factored into the score so that achieving the goal faster would generate more points.

The critical point is that every task and competence had points that were on a comparable absolute scale. A robot missing some skill could still win the competition. The point system would be published before the event so that teams knew exactly what score they could obtain (given that their robot performed as designed). This also allowed teams to make design decisions about what to implement on their robot.

One problem with the explicit quantitative scoring is trying to properly assign the proper score to the various competencies. As observed by Reid Simmons in the third competition [6], the virtual manipulation penalty [for not using real manipulation] "was much too small, providing a big disincentive for actually trying to grasp objects."

Another problem with using an explicit metric has been "gaming," where teams tailor their approaches to maximizing the metric. In some cases these high scoring entries violated the spirit of the particular competition. It was possible to exploit the metric in ways that gave less "intelligent" robots advantages in scoring.

Here is an illustrative example. Consider a "smart" robot that successfully exhibits all of the competencies; that is, it performs all of the aspects of the task itself, autonomously. The only problem is that this robot is slow, because of all of the processing. Now consider a not-quite-as-smart robot. Much less competent than the smart robot, it explicitly skips parts of the contest, gets help from the human, and consequently gets less competency points. But its so much faster that the overall total number of points is higher. In essence, speed wins even though part of why it was faster was because it skipped the slower, harder parts of the task.

To prevent these problems, it is necessary to design point systems where competencies define strict boundaries where lower level competencies cannot outscore higher-level competencies. But in complex tasks, this can become difficult to achieve.

3. Recent Years: Qualitative Metrics

In recent years, as the scope of the tasks in the contests have become more complex, we have found explicit quantitative metrics more difficult to implement, while at the same time having a desire to reduce gaming.

There are two reasons why the added complexity in the tasks have lead to difficulty. First, the tasks generally have multiple, sometimes conflicting aspects, and second, some of the required competencies are difficult to measure quantitatively themselves.

Human-robot communications is one competency that has proved difficult to judge in some domains. As observed in the second competition, "...because robots must often interact with humans, we tried to emphasize communications between man and machine. With a few exceptions, this aspect of the competition is still disappointing, and it is difficult to design tasks that reward appropriate communication."

Starting in the seventh annual competition, an hors d'oeuvres serving contest required the robots to serve conference attendees at a reception. Human-robot interaction was an important judged competency of the robots behavior, as was how much of the reception area was covered, and whether the robots could perceive when they needed to refill their trays. Obviously a single explicit quantitative metric is difficult.

Interesting, the first year of this contest, they awarded two separate awards based on different metrics. In addition to judging technical performance (which included the "intelligence," they also had a popular vote where conference attendees voted for their favorite entry. Its noteworthy that the robot that won first place in technical achievement did not win the popular vote.

We have tried various approaches that in general use more qualitative measures externally, while in some cases retaining internal quantitative metrics. In general, this means publishing more qualitative metrics, and hiding any explicit quantitative measures from the teams.

This is more of an "Olympic Figure Skating" style of scoring: a series of internal metrics are used in several categories that try to capture certain qualitative competencies. Judges, who are instructed in the qualitative aspects of these competencies, then assign a score from one to ten in each aspect, based, where possible, on an internal quantitative score. The external scores are then averaged, and each team is assigned a score from one to ten. By eliminating the external, published metrics, gaming could be avoided.

However, this style of scoring is generally difficult to implement. It also requires that the judges are carefully instructed, and that the qualitative aspects of the competencies are very well described such that teams are not mislead as to the way to achieve good scores, and to reduce the ambiguities like those that were present in the first two years of the competition.

This Olympic style of scoring is not appropriate for all competitions. For example, in the RoboCup competitions, where simulated and real robot teams compete in soccer competitions, there is a clear and natural quantitative score – the number of goals each team makes against the other.

Scoring Multiple Robots

One ongoing debate is how to measure the performance of multi-agent teams. The question is whether multi-robot entries need to exhibit better than linear improvements in performance over single robot teams.

Those who believe in super-linear improvement believe that the additional robots should introduce improvements that cannot be obtained by simply adding more robots to perform in parallel. Others believe that the proper metric involves looking at the total cost of implementing the team. Here the belief is that having multiple, inexpensive robots is equal to single expensive robots. There are several excellent articles in this proceedings on metrics for multi-agent systems.

Learning and Adaptation in Future Contests

One competency that distinguishes intelligence is the ability to learn and adapt to unanticipated events and conditions.

I would like to see competition events that require learning and adaptation in order to be most successful in the task. The learning and adaptation would not need to be directly scored, per se, but the tasks should be designed so that success is easier with those capabilities.

Although earlier competitions have stated this as a desired feature, learning usually just required building maps and learning locations of items in the environment, and adaptation was usually involved changing the robot's internal representations of the environment.

In particular, events where features of the environment change which require different sensing modalities or changes in strategies would allow for real indication of a robots "intelligence." Allowing judges to introduce failures in robots capabilities would be an ultimate test of the robots capability to adapt!

Acknowledgements

I want to thank Illah Nourbakhsh for his comments on previous AAAI contests. I also wish to thank all of the people who have contributed to past contests, either as organizers, sponsors, or as contestants.

References

[1] Arkin, R.C., 1998. "The 1997 AAAI Mobile Robot Competition and Exhibition," *AI Magazine*, 19(3), 13-17.

[2] Dean, T. and R. P. Bonasso, 1993. "1992 AAAI Robot Exhibition and Competition," *AI Magazine*, 14(1), 49-57.

[3] Hinkle, D., Kortenkamp, D., and Miller, D., 1996. "The 1995 Robot Competition and Exhibition," *AI Magazine*, 17(1), 31-45.

[4] Konolige, K., 1994. "Designing the 1993 Robot Competition," *AI Magazine*, 15(1), 57-62.

[5] Kortenkamp, D., Nourbakhsh, I., and Hinkle, D., 1997. "The 1996 AAAI Mobile Robot Competition and Exhibition," *AI Magazine*, 18(1), 25-31,1997.

[6] Simmons, R., 1995. "The 1994 AAAI Robot Competition and Exhibition," *AI Magazine*, 16(2), 19-30.

Performance Evaluation of Robotic Systems: A Proposal for a Benchmark Problem

Sunil K. Agrawal*, Armando M. Ferreira**, Stephen Pledgie**

Mechanical Engineering Department - University of Delaware

Newark DE 19716 USA - agrawal@me.udel.edu

Abstract

For highly agile autonomous systems, the dynamics plays a central role in the development of planners and feedback controllers to achieve a certain desired task. Trajectory plans that do not satisfy the system dynamics and constraints have a small likelihood for implementation without placing undue demands on the controllers. Coordinated control of such systems in groups becomes even more challenging because of the potential of dynamic interaction between members of the group, distributed nature of sensing, computation, and control. Among other desirable criteria, such as low energy consumption and constraint satisfaction, a measure of performance for robotic systems is compliance with its own dynamics and those of the other co-players in the group.

In this paper, we propose a benchmark problem for controller performance evaluation of a group of mobile robots. This benchmark experiment is inspired by a platoon of autonomous vehicles with the goal to change its formation over time. The objective is to obtain these formation changes while minimizing certain meaningful cost criteria. We assume that the physical models that describe the system are subject to errors. The sensor is not perfect and the structure of the controller has been selected by a user. For such a system, we can obtain the theoretically optimum trajectory with a measure of the cost. This cost can then be compared to the actual cost during hardware implementation on an experiment set up.

$*AssociateProfessor, **GraduateStudents$

We propose the following hardware set up with four vehicles in our Mechanical Systems Laboratory at University of Delaware. We plan to make this physical facility available to other members of the research community to test the effectiveness of their algorthims and controller implementations. Within such a facility, the different parameters of the model and controller can be altered to evaluate the performance sensitivities as a result of these change in parameters.

Our implementation on this experiment setup will be based on a two degree-of-freedom controller approach: (i) development of a reference trajectory for the system consistent with dynamics and constraints; (ii) an exponentially stable controller implemented around the reference trajectory. The reference trajectory development will be based on results from nonlinear systems theory and feedback linearization to efficiently solve the problem in a higher-order space, with a large fraction of computations done off-line ([1], [2], [3]). Such a study will bring out the issues of performance degradation during an experimental task and will provide a rich test-bed for comparing the effectiveness of different paradigms of control.

References

[1] Veeraklaew, T. and Agrawal, S. K., "A New Computation Framework for Optimization of Higher-Order Dynamic Systems", to appear in *AIAA Journal of Guidance, Control, and Dynamics*, 2000.

[2] Veeraklaew, T. and Agrawal, S. K., "Designing Robots for Optimal Performance During Repetitive Motion", *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 5, Oct 1998, 771-777.

[3] Ferreira, A. M. and Agrawal, S. K., Pledgie, S., "Planning for Autonomous Vehicle Platoons Using Differential Flatness", *to appear in Proceedings of 5th International Symposium on Advanced Vehicle Control and Control*, 2000.

A MULTI-SENSOR COOPERATIVE APPROACH FOR THE MOBILE ROBOT LOCALIZATION PROBLEM

Arnaud CLERENTIN, Laurent DELAHOCHE, Eric BRASSART
CREA
Centre de Robotique, d'Electrotechnique et d'Automatique
IUT, département Informatique, Avenue des Facultés, 80000 Amiens – France

Arnaud.Clérentin@iut.u-picardie.fr , Laurent.Delahoche@u-picardie.fr

Abstract

In this paper, we present a multi-sensor cooperation paradigm between an omnidirectional vision system and a low cost panoramic range finder system using to localize a mobile robot in its environment. These two sensors, which have been used independently until now, provide some complementary data. This association enables us to build a robust sensorial model which integrates an important number of significant primitives. We can thus realize an absolute localization of the mobile robot in particular configurations, like symmetric environments, where it is not possible to determine the position with the use of only one of the two sensors. In a first part, we present our global perception system. In a second part, we describe our sensorial model building approach and our segment classification method which takes into account the belief notion concerning a sensor. Finally we present an absolute localization method which uses three matching criteria fused thanks to the combination rules of the Dempster-Shafer theory. The basic probability assignment got for each primitive matching enables to estimate the reliability of the localization. We test our global absolute localization system on several robot's elementary moves in an indoor and symmetric environment.

1 INTRODUCTION

Autonomous mobile robots cannot rely solely on dead-reckoning to determine their configuration because dead-reckoning errors are cumulative. That's why they must use exteroceptive sensors that get information from the environment in order to estimate the robot's location more accurately. This leads to a classical localization method based on the fusion of dead reckoning data and exteroceptive data. The fusion method generally used is based on the extended kalman filter (EKF). The perception systems used both with the dead reckoning can be of different natures: a goniometer [3], the SYCLOP system [4], a laser range scanner [2].

Another approach consists in using only exteroceptive data: the robot's configuration is calculated in the environment reference without using previous information. To answer to this problem, two strategies are generally used. The first consists in marking the robot's evolution world with artificial beacons [5]. The second one consists in using the intrinsic features of the environment (doors, edges, corners…)[4] [1].

Artificial beacons can be detected fast and reliably and provide accurate positioning information with minimal processing. This kind of system is generally employed for industrial applications [10]. Unfortunately, these methods lack flexibility and modularity because it is necessary to fit out the robot's evolution environment.

The other solution consists in referencing on environment characteristic elements and offers a great modularity because the robot can localize itself directly in accordance with the landmarks. This kind of localization is founded on a matching stage between a sensorial model and a theoretical map of the environment. The perception systems used in that case are often the vision systems and the range finding ones. Perez in [6] determines with a panoramic laser range finder the absolute position of its robot by using the line segments as sensorial primitives. Similarly Yagi uses an omnidirectional vision system to develop navigation and environment map building methods [1]. We can notice that the robustness of these methods is mainly linked to the matching stage. The more precise and rich information the sensorial model will give, the more robust the matching stage will be.

That is why we have worked on a localization approach based on the cooperation of two omnidirectional perception systems: the vision system SYCLOP and a low cost range finder system. The association of these two kinds of complementary information permits to generate a sensorial model with a high descriptive level. Then, the matching stage provides an unique solution and we obtain a robust absolute determination of the robot's configuration.

The first part of this paper presents the global omnidirectional perception system. The second part deals with the sensorial model building method based on the management of two types of information. We describe also our classification method of the obtained segments on two classes according to their reliability. Our absolute localization method, based on a Dempster-Shafer multicriteria fusion approach, will be presented in the last part. In the conclusion we will analyze the experimental results reached with our mobile robot SARAH.

2 THE GLOBAL OMNIDIRECTIONAL PERCEPTION SYSTEM

To localize our mobile robot, we use an original perception system making cooperate two omnidirectional sensors: an omnidirectional vision system (SYCLOP) [4] and a low cost and fast panoramic range finder system (Figure 1). These two sensors have been developed and used independently within our laboratory [4] [9]. The rotation axis of the laser is in line with the center of the conic reflector. This geometric constraint is taken into account at the time of a previous phase of calibration.

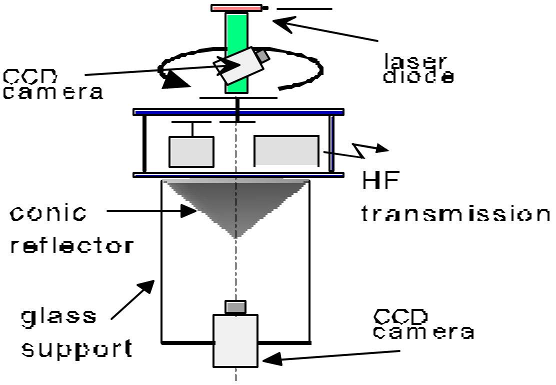


Figure 1: The global perception system

The range finder system is an active vision sensor. This method consists in projecting on the scene a visible light with known pattern geometry (a laser spot in our case). A camera images the illuminated scene with a given parallax. The desired 3D-information can be deduced from the position of the imaged laser point and the lateral distance between the projector and the camera (Figure 2).

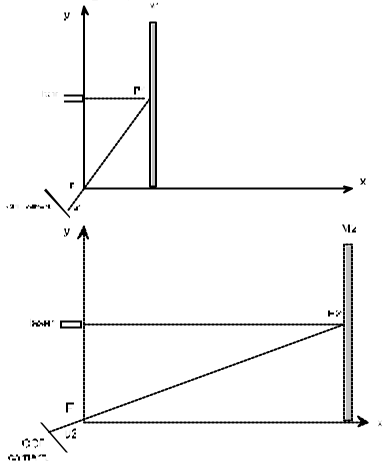


Figure 2: The geometric configuration of an active triangulation system.

The laser beam intersects the landmark M1 in the point P1 (Figure 2). This point is projected on the retinal plane through the focal point F to a point u1. A landmark M2, located at an other distance, generates a point u2. The distance of the landmark or the object Mi can be determined from the position of the point ui.

This perception system allows to obtain an omnidirectional range finding sensorial model. We manage in the sensorial model reference the cartesian distance between the laser spot and the sensor. The kind of primitives is the same that a classical range finder laser. The interest of this system is on the one hand its low cost and on the other hand its rapidity.

The prototype we built is constructed from a laser diode and a CCD camera. An infrared filter is used to extract only the light of the laser. The effective measurable distance region is designated as 0.8m-5m: this distance is thought to be a sufficient distance for a mobile robot to detect obstacles and maneuver around them.

The experimental study of this sensor is presented in [9].

The SYCLOP system, similar to the COPIS one [1], is composed of a conic mirror and a CCD camera. It allows to detect all the vertical landmarks of the environment thanks to a two dimensional projection. (Figure 3). The vertical landmarks are characterized by a radial straight line corresponding to a high contrast variation. These radial straight lines are extracted with a treatment based on the Sobel gradient. We can note that we work in fairly constraint environments, which not generate an excessive number of detected landmarks.

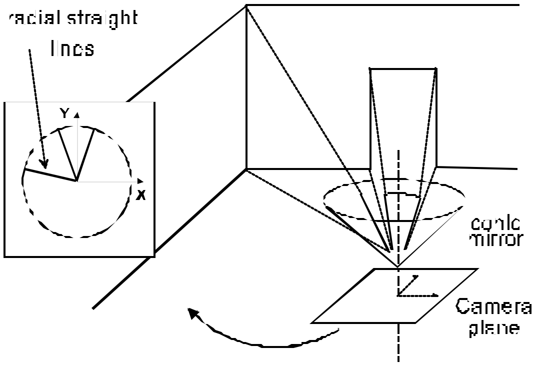


Figure 3: Principle of the omnidirectional sensor SYCLOP

This two omnidirectional sensors association permits to manage some complementary and redundant information within the same sensorial model. With the SYCLOP system we exploit, after the segmentation phase [1], the radial straight lines which characterize angles of every vertical object as, for example, doors, corners, edges, radiators. With the vision system, the information of depth cannot be gotten on an unique acquisition. For example, it is not possible to differentiate with this only sensor use the notion of opening (corridor, opening of door….) and the notion of vertical object (closed door, radiator,…) (Figure 4).

For a higher description level, it is therefore interesting to use a sensor providing some complementary information. Then we have associated to SYCLOP an inexpensive range finding sensor capable to be fast. Following a segmentation stage [9], this sensor permits us to exploit sensorial primitives that are segments (Figure 4). These segments characterize straight partitions of the environment. In this case we have the notion of depth, but it is impossible to differentiate two vertical objects placed in the same alignment: for example two closed doors placed on the same wall (Figure 4). It misses the notion of angle that will be provided by the SYCLOP system.

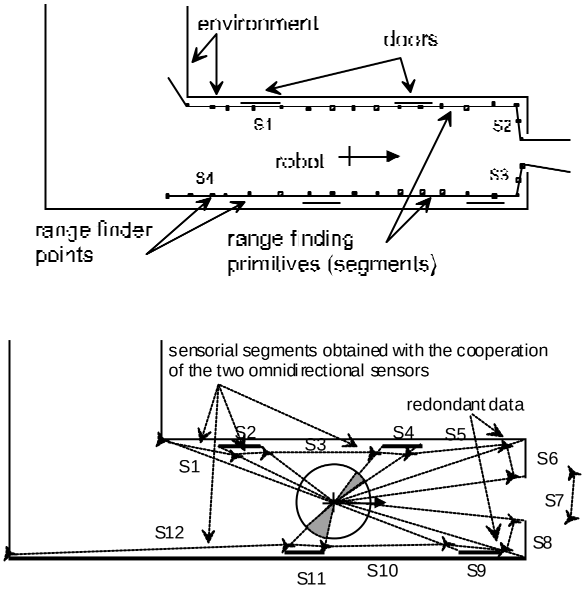

environment
doors
S1
S2
robot
S4
S3
range finder points
range finding primitives (segments)
sensorial segments obtained with the cooperation of the two omnidirectional sensors
redondant data
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12

Figure 4: Principle of the omnidirectional sensorial cooperation.

Finally this cooperative approach permits to construct a sensorial model whose descriptive level is high. This descriptive level is superior to the one obtained with each sensor individually. Moreover with an appropriated management of the redundant data (separation between two segments for the range finder and radial straight line for the SYCLOP system) we can compensate a sensorial information absence on one of this two sensors (Figure 4).

3 SENSORIAL MODEL CONSTRUCTION

The sensorial model of the evolution world is based on the taking into account of two types of data (Figure 4): the vertical landmarks angles and the segments characterizing walls. Segments are managed with two points whose coordinates are expressed in the robot's reference. The managed primitive in the final sensorial model will be segments. These segments will be determined with two types of approaches :

- An approach based on the data complementarity: this treatment consists in cutting up segments gotten with the range finder in subsegments (Figure 5). The carving is realized with the radial straight lines of the vision system.
- An approach based on the data redundancy: the redundant aspect is characterized by the detection of a vertical landmark with the two sensors (Figure 5). In certain cases a vertical landmark is detected by the range finder with the end points of segments. We will be able to confirm the existence of a segment extremity if a radial straight line corresponds to it. In case of radial straight line absence we will keep the segmentation obtained with the range finding sensorial model.

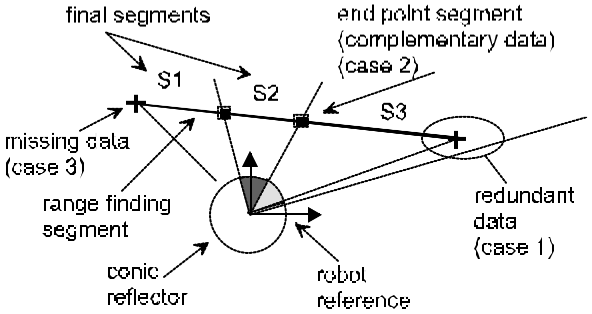


Figure 5: The Different cases of the cooperation algorithm.

We have integrated these different cases of cooperation in the sensorial model building algorithm shown on Figure 6.

The first step consists in extracting line segments from the set of points given by the sensor. We use the recursive Duda-Hart segmentation algorithm [7] [9]. To decrease the noise sensitivity of this algorithm we have added a pre-processing stage on the set of points in order to eliminate the aberrant points. Besides, in order to fit as better as possible the set of points, we apply a least square algorithm on the obtained segments.

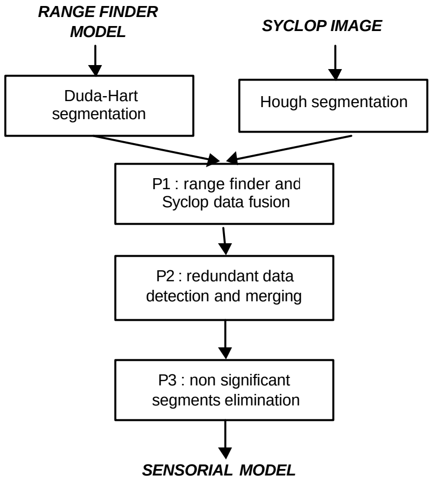

RANGE FINDER MODEL
SYCLOP IMAGE
Duda-Hart segmentation
Hough segmentation
P1 : range finder and Syclop data fusion
P2 : redundant data detection and merging
P3 : non significant segments elimination
SENSORIAL MODEL

Figure 6: Principle of the global sensorial model building algorithm

From the SYCLOP image, we treat the radial lines with a segmentation algorithm based on a simplified Hough transform. We fixed the threshold detection of a radial line (number of pixels composing a radial line) to an important value in order to keep the significant radial primitives.

The fusion step, described on Figure 4 and Figure 5, is based on the taking into account of three cases :

- The treatment of redundant data (case 1 of Figure 5). In this case we take as hypothesis to use the radial line systematically to determine the end point of a segment. The angle of a vertical landmark is determined more precisely with the vision system that with the range finder.
- The treatment of complementary data (case 2 of Figure 5). This treatment consists in cutting up a range finding segment into several final subsegments. This stage is based on the segment intersection determination.

- ❑ The treatment of missing data (case 3 of Figure 5). The notion of missing data is here characterized by a vertical landmark which is not detected with the vision sensor. In this case the range finding breakpoint is considered directly.

During this stage, we classify the segments and subsegments we get in two classes of reliability: a class "SURE" and a class "UNCERTAIN". In this purpose, we take into account five criterion for each segment.

The first criteria is the mean distance between the range finding points contained by the segment and this segment. If this mean distance is high, it means that the points are not very well aligned, so this segment is not very sure.

The second criteria is the number of points supported by the segment. This criteria is only discriminative when the segment contains very few points. In this case, it is not sure.

The third criteria is the segment density of points. As shown in [9], a major drawback of this kind of triangulation depth sensor is a decreasing resolution with increasing distance. So, this criteria, which is linked to the mean distance between the sensor and the set of point, is a good indicator of the segment reliability (more distant the set of points is, less the precision is). Considering the measure extent of the sensor (0.8m from 5m), the minimal and maximal density are as shown on Figure 7.

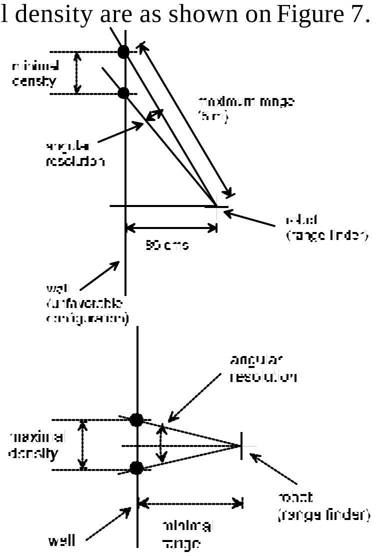


Figure 7: quantification of the density criteria

The fourth criteria analyzes if the segment has been detected by one or by the two sensors. The different cases are:

- ❑ The two extremities of a segment are detected only by the laser range finder (segment S1 in the case 1 of Figure 8). This segment has a weight of 1.
- ❑ One extremity of a segment is detected by the laser range finder and the other extremity is detected only by the conic mirror (segment S1 in the case 2 of Figure 8). This segment has a weight of 2 because, as we say before, we think that the radial straight lines are more precise and reliable.
- ❑ One extremity of a segment is detected by the two sensors and the other extremity is detected only by the laser, or the two extremities are detected only by the conic mirror (segment S1 in the case 3 of Figure 8). This segment has a weight of 3.
- ❑ One extremity of a segment is detected by the two sensors and the other extremity is detected only by SYCLOP (segment S1 in the case 4 of Figure 8). This segment has a weight of 4.
- ❑ The two extremities of a segment are detected by the two sensors (segment S1 on the case 5 of Figure 8). This segment has a weight of 5.

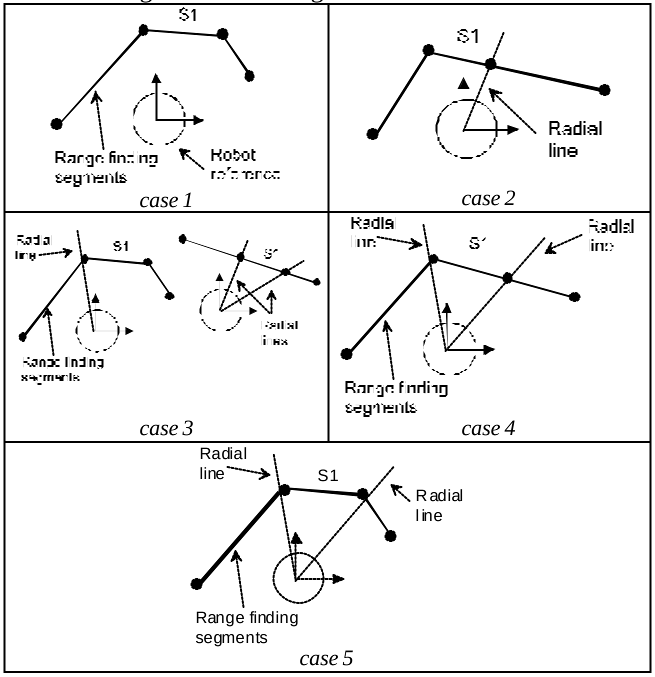


Figure 8: powdered segment, the four cases.

The last criteria concerns a gray level curves extracted from the SYCLOP image. We take into consideration five concentric gray level circles whose average is made. We obtain thus one gray level curve from 0 to 360 degrees. We apply on the portions of curve which represent a segment a least square algorithm. We obtain a straight line and we compute the mean difference of the gray level value from this line. If this mean difference is high, this means that the gray level sector is not constant. This case occurs generally when a landmark has not been detected by SYCLOP, so this segment is not sure.

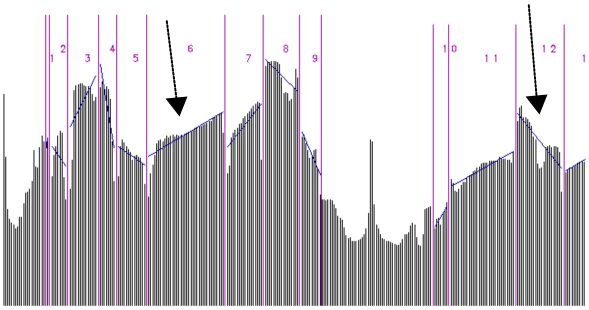


Figure 9 : the gray level curve of the experimental result shown fig. 14

The fusion of these five criteria is made thanks to the combination rules of the Dempster-Shafer theory [8][11]. We use this theory because it is an interesting formalism which enables to represent ignorance. Our frame of discernment is composed of two elements: "SURE" and "UNCERTAIN". The basic probability assignments m_1, m_2, m_3, m_4 and m_5 for this five criteria are shown in Figure 10. We can see that, for certain values, the criterion are not discriminative and Dempster-Shafer enables to represent this ignorance (for example, if the density is equal to 0.12 points/cm, this value does not permit to take a decision SURE or UNCERTAIN for this criteria).

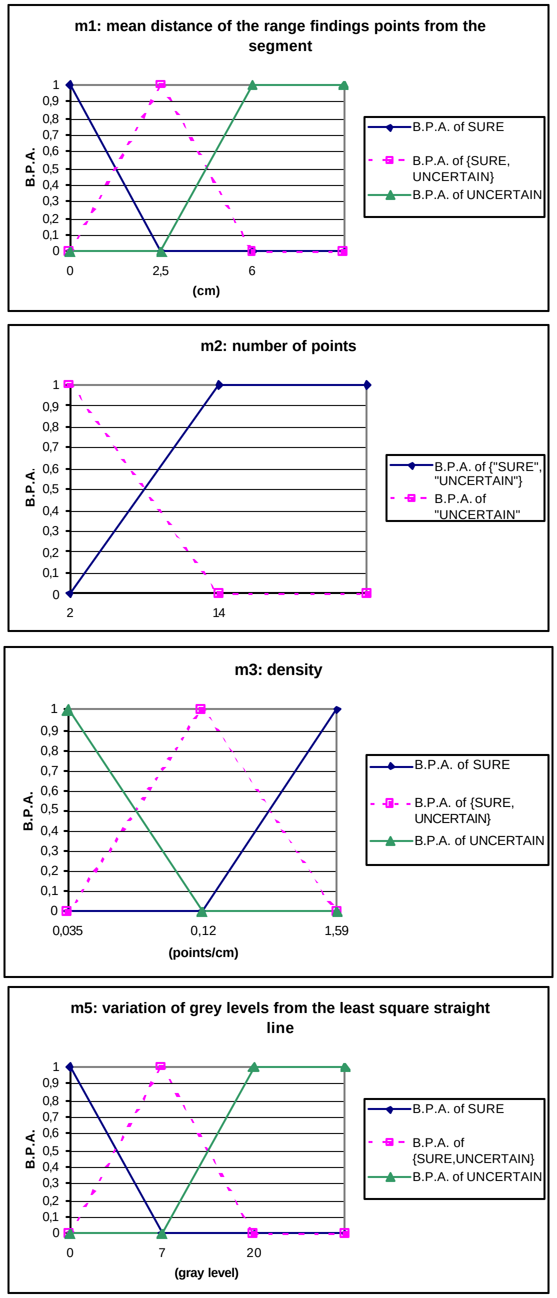


Figure 10: the B.P.As of the four classification criteria ({SURE,UNCERTAIN}=Q)

For the fourth criteria, the B.P.A. are:

weight 1:m_4(SURE)=m_4(Θ)=0.6, m_4(UNCERTAIN)=0.4
weight 2:m_4(SURE)=0, m_4(Θ)=1, m_4(UNCERTAIN)=0
weight 3:m_4(SURE)=0.3,m_4(Θ)=0.7,m_4(UNCERTAIN)=0
weight 4:m_4(SURE)=0.6,m_4(Θ)=0.4,m_4(UNCERTAIN)=0
weight 5:m_4(SURE)=1, m_4(Θ)=m_4(UNCERTAIN)=0

We can then perform the combination calculation thanks to the Dempster-Shafer rules [8][11]. If the conflict coefficient k between the elements of the frame of discernment is superior to 0.7, it means that our criteria disagree. In this case, we decided that our segment is uncertain. If $k<0.7$, we compute the combination of belief functions for each element of the frame of discernment and we choose the class which has the maximal B.P.A.

The last stage (P3 on Figure 6) consists in eliminating the non significant segments in the final cooperative sensorial model. A non significant segment is characterized by a number of range finding points equal to 0 and a length (Cartesian distance) inferior to a predetermined threshold.

This stage permits to decrease the combinatory aspect of the matching stage and to increase the robustness.

This building algorithm enables to get a sensorial model where the number of exploitable primitives is more important than the number of primitives got by each sensor when it works individually. Besides, we obtain a certainty information of a segment by considering five criteria. This information will be used in the matching phase.

4 ABSOLUTE LOCALIZATION METHOD

The robot configuration is determined by matching the sensorial model, got by multisensor cooperation, with a theoretical map of the environment. The primitives used for this matching phase are segments. Therefore, all environment’s elements like doors, walls, windows, radiators… are indexed as segments in the theoretical map.

For each segment, we have considered three correspondence tests, which are similar to these used by Crowley [7]:

❑ the angular difference between the two segments,
❑ the difference in length between the two segments,
❑ the distance between the centers of the two segments.

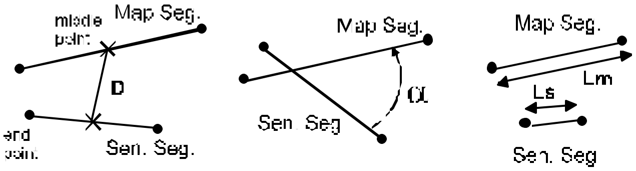


Figure 11: The three matching criteria.

The fusion of these three treatments is made thanks to the combination rules of the Dempster-Shafer theory [8]. Our frame of discernment is composed of two elements: YES and NO corresponding to those assertions : "Yes, we can match the two segments" and "No, we can not match the two segments". For each criterion, we have determined the Basic Probability Assignments (B.P.A.) m_1, m_2 , m_3 shown Figure 12.

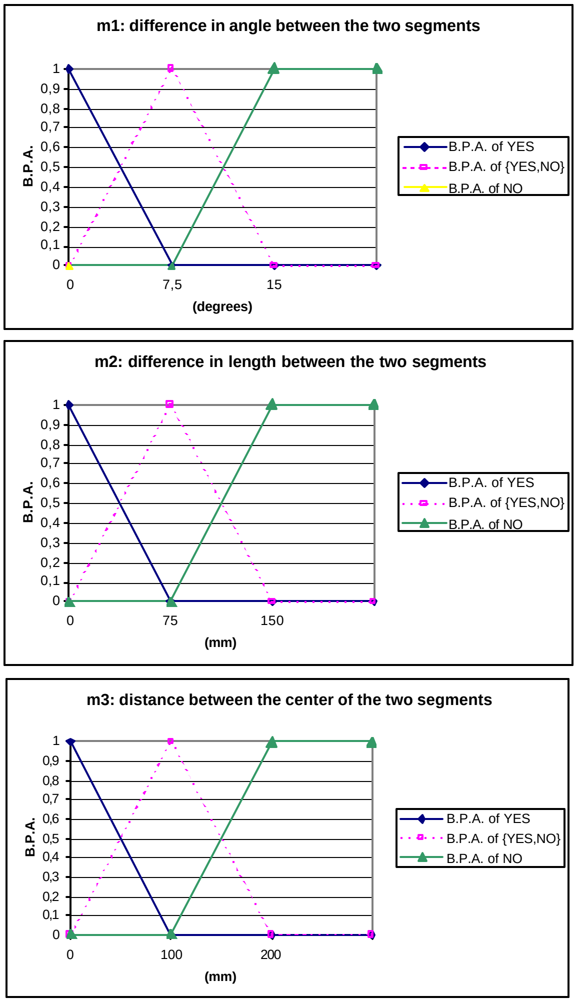


Figure 12: basic probability assignments of matching criteria ({YES,NO}=Q)

We can then perform the combination calculation thanks to the Dempster-Shafer rules [8]. Since we have three criteria, we first fuse the two first criteria.

The conflict coefficient between these two first criteria is:

$$k_{12} = m_1(YES).m_2(NO) + m_1(NO).m_2(YES) \qquad (1)$$

If k_{12}<1, the conflict is not complete and the combination of belief functions m_{12} for these two elements of the frame of discernment is given by:

$$m_{12}\ (YES) = \frac{m_1(YES).m_2(YES) + m_1(YES).m_2(\Theta) + m_1(\Theta).m_2(YES)}{1-k_{12}}$$

$$m_{12}(NO) = \frac{m_1(NO).m_2(NO) + m_1(NO).m_2(\Theta) + m_1(\Theta).m_2(NO)}{1-k_{12}}$$

$$m_{12}(\Theta) = \frac{m_1(\Theta).m_2(\Theta)}{1-k_{12}} \qquad (2)$$

Then we fuse the last criterion. We compute the conflict coefficient k (3) between this criterion and the two criteria we have fused above:

$$k = m_{12}(YES).m_3(NO) + m_{12}(NO).m_2(YES) \qquad (3)$$

If k>0.7, we think that the conflict is too high. So we decide to take a prudent decision: we don't match the two segments. If k<0.7, we compute the combination of belief functions for each focal element:

$$m(YES) = \frac{m_{12}(YES).m_3(YES) + m_{12}(YES).m_3(\Theta) + m_{12}(\Theta).m_3(YES)}{1-k}$$

$$m(NO) = \frac{m_{12}(NO).m_3(NO) + m_{12}(NO).m_3(\Theta) + m_{12}(\Theta).m_3(NO)}{1-k}$$

$$m(\Theta) = \frac{m_{12}(\Theta).m_3(b)}{1-k} = \frac{m_1(\Theta).m_2(\Theta).m_3(\Theta)}{1-k} \qquad (4)$$

The segments are matched if B.P.A. for the *YES m(YES)* is superior to the B.P.A. for the *NO m(NO)*.

The first stage of this localization algorithm consists in determining a list of sensorial segments *Ls* which have a strong probability of existence. This segments are the "SURE" segments obtained during the fusion stage.

We consider that the length of these segments has been determined with a good accuracy. So, our starting correspondence test is the length of a segment.

In the second stage, we consider a segment Ls_k from the list *Ls* and we search the theoretical map segments which length is similar to the Ls_k segment length. Each found theoretical segment is superposed on the sensorial segment Ls_k and we apply the third step in order to test the correspondence of the other sensorial segments.

The third step consists in applying the three criteria describe above on all the segments on the list *Ls* except the segment Ls_k. A segment is matched if the B.P.A. for the *YES* is superior to the B.P.A. for the *NO*. To choice the optimal matching solution we calculate a *V* criteria. For each matched segment pair, we increment this *V* coefficient which characterizes the robustness of the global matching. *V* is managed with the following algorithm:

```
Given:
-   B the B.P.A. for the YES of the matched
    segment pair
-   W a weight linked to the segment's class
    (SURE segment: w=3, UNCERTAIN segment:
    w=1).
FOR each global matching
      V=0
      FOR each segment matched
            V = V + (B*W)
      END
END
```

So we can see that *V* is an interesting and discriminative indicator of the global matching relevance since *V* takes into account the class of each matched segment ("SURE", "UNCERTAIN") and the quality of each matched pair (through the B.P.A. for the *YES*).

These three steps are then repeated for all the *Ls* list segments. The final solution is the one which permits the maximal *V*.

5 EXPERIMENTAL RESULTS

To test the robustness of our localization algorithm, we have performed it on several sensorial acquisitions made in an indoor environment (Figure 13). The two omnidirectional acquisitions are made when the robot is stopped. The omnidirectional acquisitions and the localization algorithm are computed in a Pentium PC located on our mobile robot. A Matrox Meteor II video card is used to acquire the omnidirectionnal image and the laser acquisition. Our experimental perception system is shown on Figure 13.

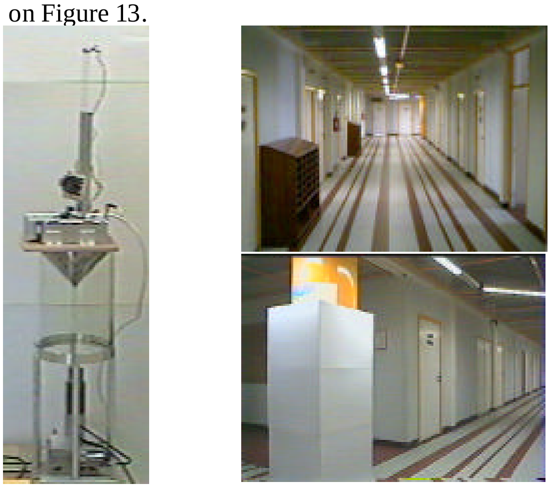
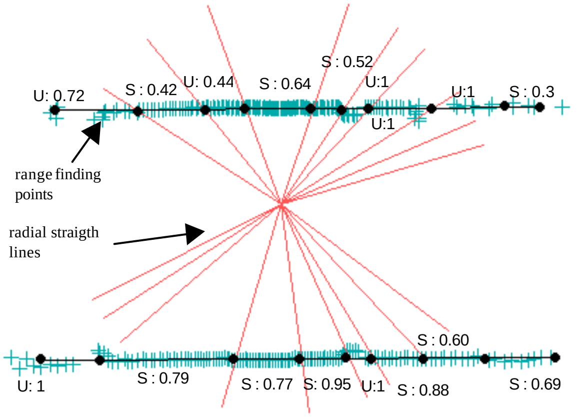


Figure 13: Our global omnidirectional perception system and the experimental indoor environment.

In order to show the interest of our cooperative approach, we have tested our localization method on symmetric environments (the first picture on Figure 13). The use of one sensor individually instead of the two sensors emphasizes the robustness problem: a strong failure rate has been observed for the matching phase when we use only one sensor [9].

The first environment is a long corridor (length: 50 meters). Figure 14 shows a sensorial model got with our cooperative approach. The robot is located in the middle of the corridor (Figure 13). We can see on Figure 14 the final decomposition on an set of segments which represent doors and parts of wall. We show on this figure the radial straight lines obtained with the omnidirectional conic mirror. We must note that, for this environment, the depth sensor would not have been able to localize the robot: two parallel identical segments would have been detected. The SYCLOP system used alone would have posed the problem of environment symmetry. We can also remark that uncertain segments are the segments which are far from the robot (not well aligned) or which correspond to the pillars of the corridor (not detected during the Duda-Hart segmentation stage). The robot final position successfully obtained shows the robustness of our method and its accuracy. We have indeed a position error of 8cm and an orientation error of 3 degree.



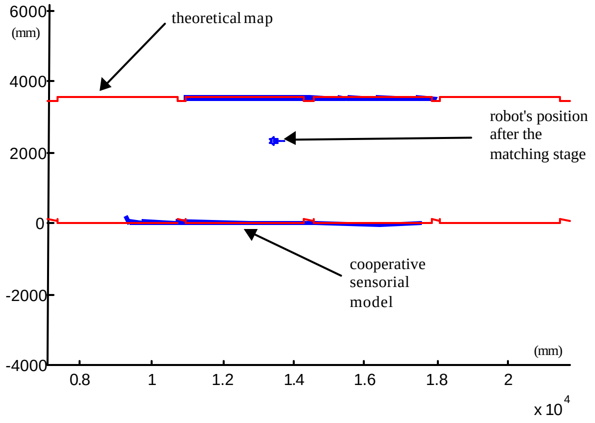


Figure 14: the cooperative sensorial model with the segments classification and the BPA(U=UNCERTAIN, S=SURE) (first figure) and the final position determination corresponding to the optimal matching

We show on Figure 15 results obtained in an other symmetric environment: a laboratory square hall.

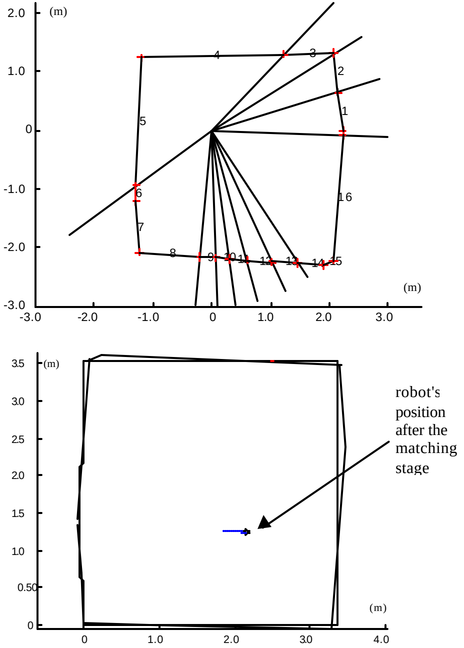


Figure 15: cooperative sensorial model and final position determination in a hall environment.

The same remarks can be done: the use of the two sensors provides enough sensorial information to enable the matching algorithm to converge to a coherent solution.

The third environment is the end of the corridor shown Figure 13. This environment constitutes a favorable experimental configuration: it is not symmetric and it has an important number of exploitable landmarks (figure 10). We can note here on several robot's configuration determination that our matching selection criteria is highly discriminative: the good configuration has been computed on all the acquisitions.

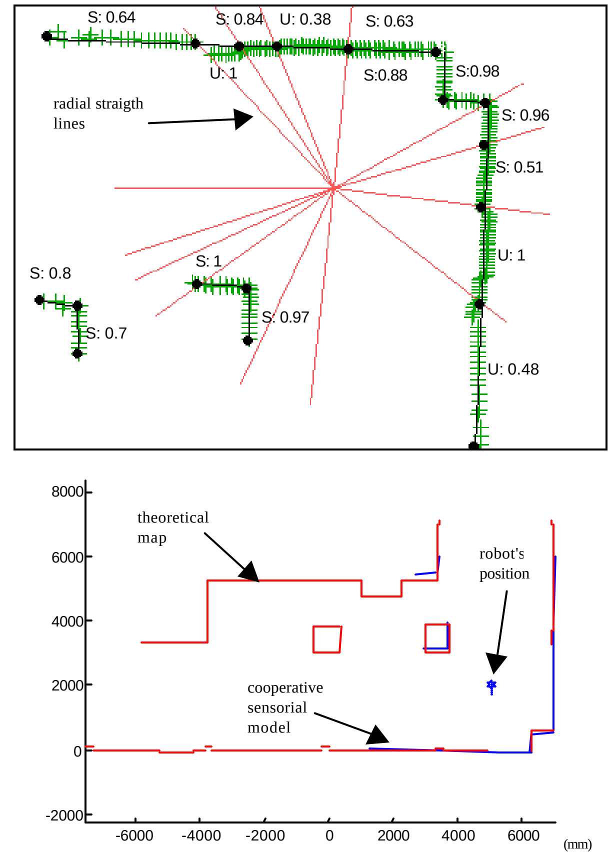


Figure 16: cooperative sensorial model (first figure) and final position determination.

Finally on a complete path makes in the corridor by our robot mobile SARAH, we could note on 40 acquisitions that, on the one hand, all the absolute configurations have been determined correctly, and, on the other hand, the mean error was equal to 11 cm in position and 3 degree in orientation.

In spite of an important combinatory aspect, our cooperative localization method proves to be robust and particularly accurate.

6 Conclusion

We have presented in this study an absolute localization approach based on the cooperation between two omnidirectional sensors: an omnidirectionnal vision sensor and a range finding sensor. This association allows to treat two types of complementary data. Then we obtain a highly descriptive sensorial model which integrates an important number of primitives and enables to increase the robustness of the matching stage. We classify also every sensed segment in two reliability classes according to five criteria fused thanks to the Dempster-Shafer rules. The absolute localization paradigm based on this matching stage takes into account several criteria which are merged with the Dempster Shafer rules. The choice of the optimal matching is based on a highly discriminative criteria which associates the segment reliability classes and a B.P.A. linked to the matching stage. We have tested our cooperative absolute localization algorithm on several particular environment like for example symmetrical environment. On the one hand, we can note on these experimental results that the robot's configuration determination is realized in a unique way and on the other hand the absolute robot's configuration is calculated with a relatively weak systematic error.

References

[1] Y. Yagi, Y. Nishizawa, M. Yachida, "Map-based navigation for a mobile robot with omnidirectional image sensor COPIS", IEEE Trans. on Robotics and Automation Vol. 11, pp. 634-648, October 1995.

[2] J.Gomes-Mota, M.I. Ribeiro, "A multi-layer robot localisation solution using a laser scanner on reconstructed 3D models", Proc. on the 6th Int. Symposium on Intelligent Robotic Systems, Scotland, 1998.

[3] P. Bonnifait, G. Garcia, "Design and Experimental Validation of an Odometric and Goniometric Localization System for Outdoor Robot Vehicles.", IEEE Trans. on Rob. and Aut. Vol. 14, No 4, pp. 541-548, August 1998.

[4] C. Drocourt, L. Delahoche, C. Pegard, C. Cauchois , "Localization method based on omnidirectional stereoscopic vision and dead-reckoning", Proc. of the IEEE Int. Conf. on Int. Robots and Systems, Korea, October 1999

[5] H.R Beom, H.S. Cho, "Mobile robot localization using a single rotating sonar and two passive cylindrical beacons", Robotica, Vol. 13, pp. 243-252, 1995.

[6] J.A. Perez, J.A. Castellanos, J.M.M. Montiel, "Continuous localization: vision vs. laser", IEEE Proc. of Int. Conf. on Rob. and Aut. pp 2917-2923, Detroit, May 1999.

[7] J. Crowley, "Navigation for an intelligent mobile robot", IEEE Journal on Robotics and Automation, Vol. RA-1, n°1, pp. 31-41, March 1985.

[8] G.A. Shafer, "A mathematical theory of evidence", Princeton : university press, 1976.

[9] A. Clérentin, C. Pégard, C. Drocourt "Environment Exploration Using an Active Vision Sensor", Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS'99), Korea ,October 1999.

[10] J. Hollingum, "Caterpillar make the earth move : automatically", Industrial Robot, Vol. 18, N° 2, pp. 15-18, 1991.

[11] A. Dempster, "Upper and lower probabilities induced by a multivalued mapping", Annals of mathematical statistics 38:325-339, 1967.

A MULTI-SENSOR COOPERATIVE APPROACH FOR THE MOBILE ROBOT LOCALIZATION PROBLEM

Arnaud CLERENTIN, Laurent DELAHOCHE, Eric BRASSART
CREA
Centre de Robotique, d'Electrotechnique et d'Automatique
IUT, département Informatique, Avenue des Facultés, 80000 Amiens – France

Arnaud.Clérentin@iut.u-picardie.fr , Laurent.Delahoche@u-picardie.fr

Abstract

In this paper, we present a multi-sensor cooperation paradigm between an omnidirectional vision system and a low cost panoramic range finder system using to localize a mobile robot in its environment. These two sensors, which have been used independently until now, provide some complementary data. This association enables us to build a robust sensorial model which integrates an important number of significant primitives. We can thus realize an absolute localization of the mobile robot in particular configurations, like symmetric environments, where it is not possible to determine the position with the use of only one of the two sensors. In a first part, we present our global perception system. In a second part, we describe our sensorial model building approach and our segment classification method which takes into account the belief notion concerning a sensor. Finally we present an absolute localization method which uses three matching criteria fused thanks to the combination rules of the Dempster-Shafer theory. The basic probability assignment got for each primitive matching enables to estimate the reliability of the localization. We test our global absolute localization system on several robot's elementary moves in an indoor and symmetric environment.

1 INTRODUCTION

Autonomous mobile robots cannot rely solely on dead-reckoning to determine their configuration because dead-reckoning errors are cumulative. That's why they must use exteroceptive sensors that get information from the environment in order to estimate the robot's location more accurately. This leads to a classical localization method based on the fusion of dead reckoning data and exteroceptive data. The fusion method generally used is based on the extended kalman filter (EKF). The perception systems used both with the dead reckoning can be of different natures: a goniometer [3], the SYCLOP system [4], a laser range scanner [2].

Another approach consists in using only exteroceptive data: the robot's configuration is calculated in the environment reference without using previous information. To answer to this problem, two strategies are generally used. The first consists in marking the robot's evolution world with artificial beacons [5]. The second one consists in using the intrinsic features of the environment (doors, edges, corners…)[4] [1].

Artificial beacons can be detected fast and reliably and provide accurate positioning information with minimal processing. This kind of system is generally employed for industrial applications [10]. Unfortunately, these methods lack flexibility and modularity because it is necessary to fit out the robot's evolution environment.

The other solution consists in referencing on environment characteristic elements and offers a great modularity because the robot can localize itself directly in accordance with the landmarks. This kind of localization is founded on a matching stage between a sensorial model and a theoretical map of the environment. The perception systems used in that case are often the vision systems and the range finding ones. Perez in [6] determines with a panoramic laser range finder the absolute position of its robot by using the line segments as sensorial primitives. Similarly Yagi uses an omnidirectional vision system to develop navigation and environment map building methods [1]. We can notice that the robustness of these methods is mainly linked to the matching stage. The more precise and rich information the sensorial model will give, the more robust the matching stage will be.

That is why we have worked on a localization approach based on the cooperation of two omnidirectional perception systems: the vision system SYCLOP and a low cost range finder system. The association of these two kinds of complementary information permits to generate a sensorial model with a high descriptive level. Then, the matching stage provides an unique solution and we obtain a robust absolute determination of the robot's configuration.

The first part of this paper presents the global omnidirectional perception system. The second part deals with the sensorial model building method based on the management of two types of information. We describe also our classification method of the obtained segments on two classes according to their reliability. Our absolute localization method, based on a Dempster-Shafer multicriteria fusion approach, will be presented in the last part. In the conclusion we will analyze the experimental results reached with our mobile robot SARAH.

2 THE GLOBAL OMNIDIRECTIONAL PERCEPTION SYSTEM

To localize our mobile robot, we use an original perception system making cooperate two omnidirectional sensors: an omnidirectional vision system (SYCLOP) [4] and a low cost and fast panoramic range finder system (Figure 1). These two sensors have been developed and used independently within our laboratory [4] [9]. The rotation axis of the laser is in line with the center of the conic reflector. This geometric constraint is taken into account at the time of a previous phase of calibration.

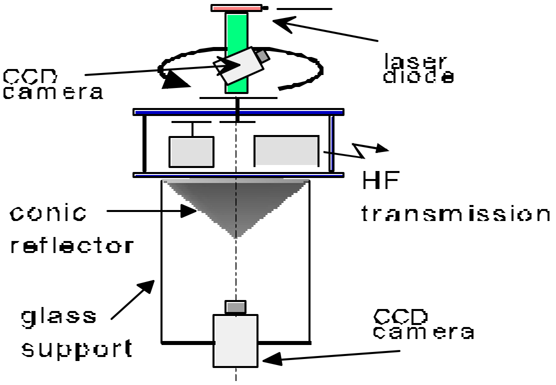


Figure 1: The global perception system

The range finder system is an active vision sensor. This method consists in projecting on the scene a visible light with known pattern geometry (a laser spot in our case). A camera images the illuminated scene with a given parallax. The desired 3D-information can be deduced from the position of the imaged laser point and the lateral distance between the projector and the camera (Figure 2).

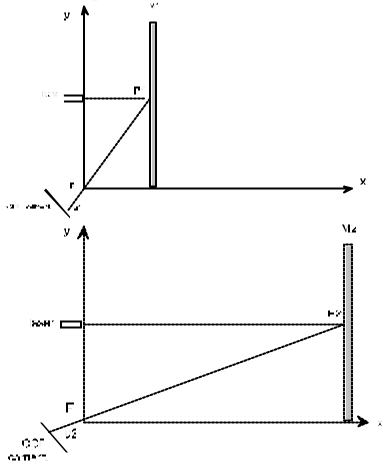


Figure 2: The geometric configuration of an active triangulation system.

The laser beam intersects the landmark M1 in the point P1 (Figure 2). This point is projected on the retinal plane through the focal point F to a point u1. A landmark M2, located at an other distance, generates a point u2. The distance of the landmark or the object Mi can be determined from the position of the point ui.

This perception system allows to obtain an omnidirectional range finding sensorial model. We manage in the sensorial model reference the cartesian distance between the laser spot and the sensor. The kind of primitives is the same that a classical range finder laser. The interest of this system is on the one hand its low cost and on the other hand its rapidity.

The prototype we built is constructed from a laser diode and a CCD camera. An infrared filter is used to extract only the light of the laser. The effective measurable distance region is designated as 0.8m-5m: this distance is thought to be a sufficient distance for a mobile robot to detect obstacles and maneuver around them.

The experimental study of this sensor is presented in [9].

The SYCLOP system, similar to the COPIS one [1], is composed of a conic mirror and a CCD camera. It allows to detect all the vertical landmarks of the environment thanks to a two dimensional projection. (Figure 3). The vertical landmarks are characterized by a radial straight line corresponding to a high contrast variation. These radial straight lines are extracted with a treatment based on the Sobel gradient. We can note that we work in fairly constraint environments, which not generate an excessive number of detected landmarks.

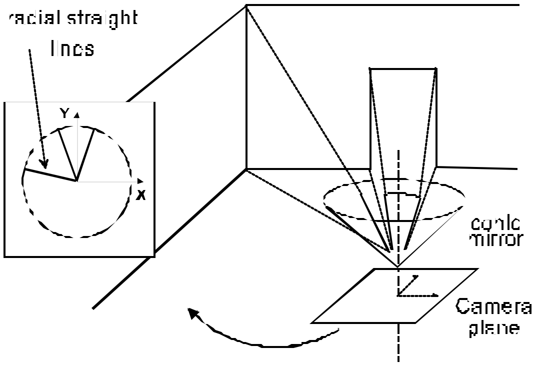


Figure 3: Principle of the omnidirectional sensor SYCLOP

This two omnidirectional sensors association permits to manage some complementary and redundant information within the same sensorial model. With the SYCLOP system we exploit, after the segmentation phase [1], the radial straight lines which characterize angles of every vertical object as, for example, doors, corners, edges, radiators. With the vision system, the information of depth cannot be gotten on an unique acquisition. For example, it is not possible to differentiate with this only sensor use the notion of opening (corridor, opening of door….) and the notion of vertical object (closed door, radiator,…) (Figure 4).

For a higher description level, it is therefore interesting to use a sensor providing some complementary information. Then we have associated to SYCLOP an inexpensive range finding sensor capable to be fast. Following a segmentation stage [9], this sensor permits us to exploit sensorial primitives that are segments (Figure 4). These segments characterize straight partitions of the environment. In this case we have the notion of depth, but it is impossible to differentiate two vertical objects placed in the same alignment: for example two closed doors placed on the same wall (Figure 4). It misses the notion of angle that will be provided by the SYCLOP system.

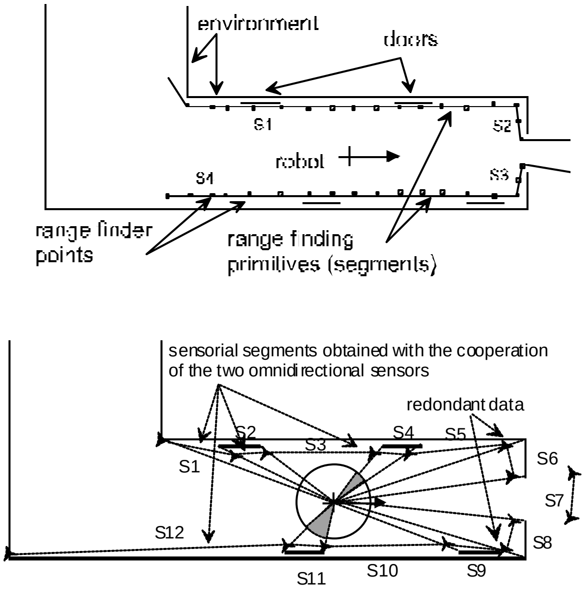

environment
doors
S1
S2
robot
S4
S3
range finder points
range finding primitives (segments)
sensorial segments obtained with the cooperation of the two omnidirectional sensors
redondant data
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12

Figure 4: Principle of the omnidirectional sensorial cooperation.

Finally this cooperative approach permits to construct a sensorial model whose descriptive level is high. This descriptive level is superior to the one obtained with each sensor individually. Moreover with an appropriated management of the redundant data (separation between two segments for the range finder and radial straight line for the SYCLOP system) we can compensate a sensorial information absence on one of this two sensors (Figure 4).

3 SENSORIAL MODEL CONSTRUCTION

The sensorial model of the evolution world is based on the taking into account of two types of data (Figure 4): the vertical landmarks angles and the segments characterizing walls. Segments are managed with two points whose coordinates are expressed in the robot's reference. The managed primitive in the final sensorial model will be segments. These segments will be determined with two types of approaches :

- ❑ An approach based on the data complementarity: this treatment consists in cutting up segments gotten with the range finder in subsegments (Figure 5). The carving is realized with the radial straight lines of the vision system.
- ❑ An approach based on the data redundancy: the redundant aspect is characterized by the detection of a vertical landmark with the two sensors (Figure 5). In certain cases a vertical landmark is detected by the range finder with the end points of segments. We will be able to confirm the existence of a segment extremity if a radial straight line corresponds to it. In case of radial straight line absence we will keep the segmentation obtained with the range finding sensorial model.

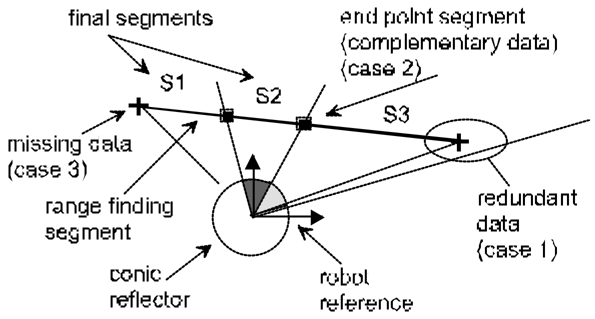


Figure 5: The Different cases of the cooperation algorithm.

We have integrated these different cases of cooperation in the sensorial model building algorithm shown on Figure 6.

The first step consists in extracting line segments from the set of points given by the sensor. We use the recursive Duda-Hart segmentation algorithm [7] [9]. To decrease the noise sensitivity of this algorithm we have added a pre-processing stage on the set of points in order to eliminate the aberrant points. Besides, in order to fit as better as possible the set of points, we apply a least square algorithm on the obtained segments.

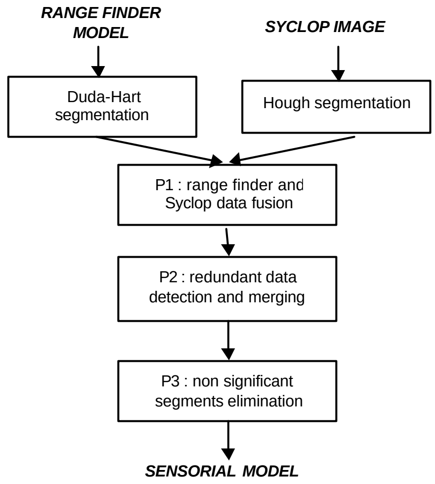

RANGE FINDER MODEL
SYCLOP IMAGE
Duda-Hart segmentation
Hough segmentation
P1 : range finder and Syclop data fusion
P2 : redundant data detection and merging
P3 : non significant segments elimination
SENSORIAL MODEL

Figure 6: Principle of the global sensorial model building algorithm

From the SYCLOP image, we treat the radial lines with a segmentation algorithm based on a simplified Hough transform. We fixed the threshold detection of a radial line (number of pixels composing a radial line) to an important value in order to keep the significant radial primitives.

The fusion step, described on Figure 4 and Figure 5, is based on the taking into account of three cases :

- ❑ The treatment of redundant data (case 1 of Figure 5). In this case we take as hypothesis to use the radial line systematically to determine the end point of a segment. The angle of a vertical landmark is determined more precisely with the vision system that with the range finder.
- ❑ The treatment of complementary data (case 2 of Figure 5). This treatment consists in cutting up a range finding segment into several final subsegments. This stage is based on the segment intersection determination.

- ❑ The treatment of missing data (case 3 of Figure 5). The notion of missing data is here characterized by a vertical landmark which is not detected with the vision sensor. In this case the range finding breakpoint is considered directly.

During this stage, we classify the segments and subsegments we get in two classes of reliability: a class "SURE" and a class "UNCERTAIN". In this purpose, we take into account five criterion for each segment.

The first criteria is the mean distance between the range finding points contained by the segment and this segment. If this mean distance is high, it means that the points are not very well aligned, so this segment is not very sure.

The second criteria is the number of points supported by the segment. This criteria is only discriminative when the segment contains very few points. In this case, it is not sure.

The third criteria is the segment density of points. As shown in [9], a major drawback of this kind of triangulation depth sensor is a decreasing resolution with increasing distance. So, this criteria, which is linked to the mean distance between the sensor and the set of point, is a good indicator of the segment reliability (more distant the set of points is, less the precision is). Considering the measure extent of the sensor (0.8m from 5m), the minimal and maximal density are as shown on Figure 7.

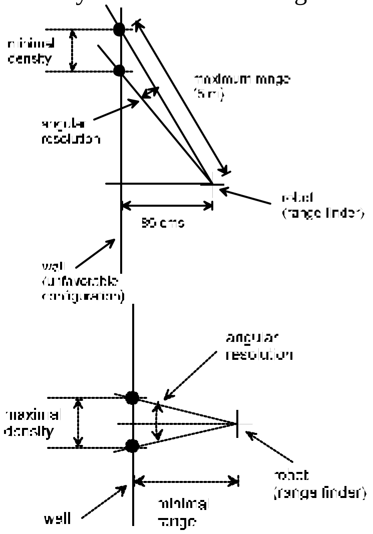


Figure 7: quantification of the density criteria

The fourth criteria analyzes if the segment has been detected by one or by the two sensors. The different cases are:

- ❑ The two extremities of a segment are detected only by the laser range finder (segment S1 in the case 1 of Figure 8). This segment has a weight of 1.
- ❑ One extremity of a segment is detected by the laser range finder and the other extremity is detected only by the conic mirror (segment S1 in the case 2 of Figure 8). This segment has a weight of 2 because, as we say before, we think that the radial straight lines are more precise and reliable.
- ❑ One extremity of a segment is detected by the two sensors and the other extremity is detected only by the laser, or the two extremities are detected only by the conic mirror (segment S1 in the case 3 of Figure 8). This segment has a weight of 3.
- ❑ One extremity of a segment is detected by the two sensors and the other extremity is detected only by SYCLOP (segment S1 in the case 4 of Figure 8). This segment has a weight of 4.
- ❑ The two extremities of a segment are detected by the two sensors (segment S1 on the case 5 of Figure 8). This segment has a weight of 5.

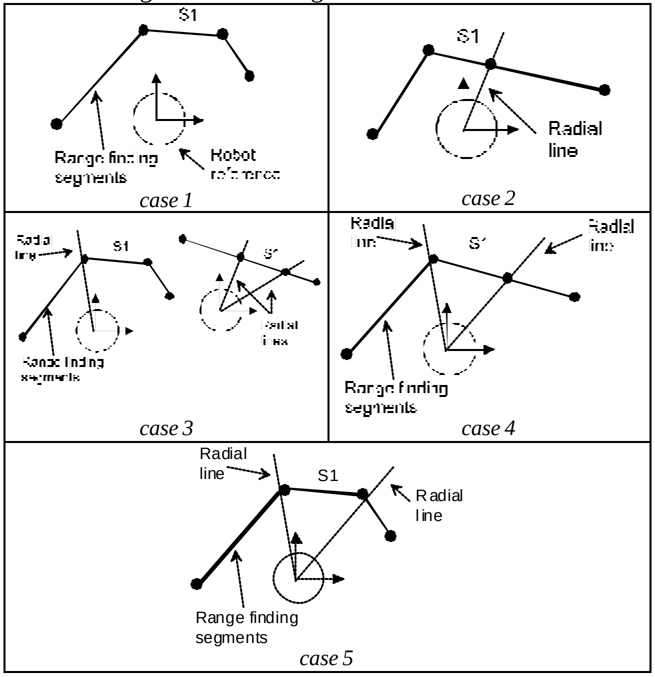


Figure 8: powdered segment, the four cases.

The last criteria concerns a gray level curves extracted from the SYCLOP image. We take into consideration five concentric gray level circles whose average is made. We obtain thus one gray level curve from 0 to 360 degrees. We apply on the portions of curve which represent a segment a least square algorithm. We obtain a straight line and we compute the mean difference of the gray level value from this line. If this mean difference is high, this means that the gray level sector is not constant. This case occurs generally when a landmark has not been detected by SYCLOP, so this segment is not sure.

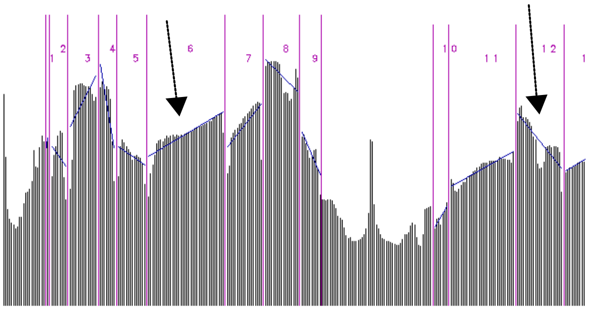


Figure 9 : the gray level curve of the experimental result shown fig. 14

The fusion of these five criteria is made thanks to the combination rules of the Dempster-Shafer theory [8][11]. We use this theory because it is an interesting formalism which enables to represent ignorance. Our frame of discernment is composed of two elements: "SURE" and "UNCERTAIN". The basic probability assignments m_1, m_2, m_3, m_4 and m_5 for this five criteria are shown in Figure 10. We can see that, for certain values, the criterion are not discriminative and Dempster-Shafer enables to represent this ignorance (for example, if the density is equal to 0.12 points/cm, this value does not permit to take a decision SURE or UNCERTAIN for this criteria).

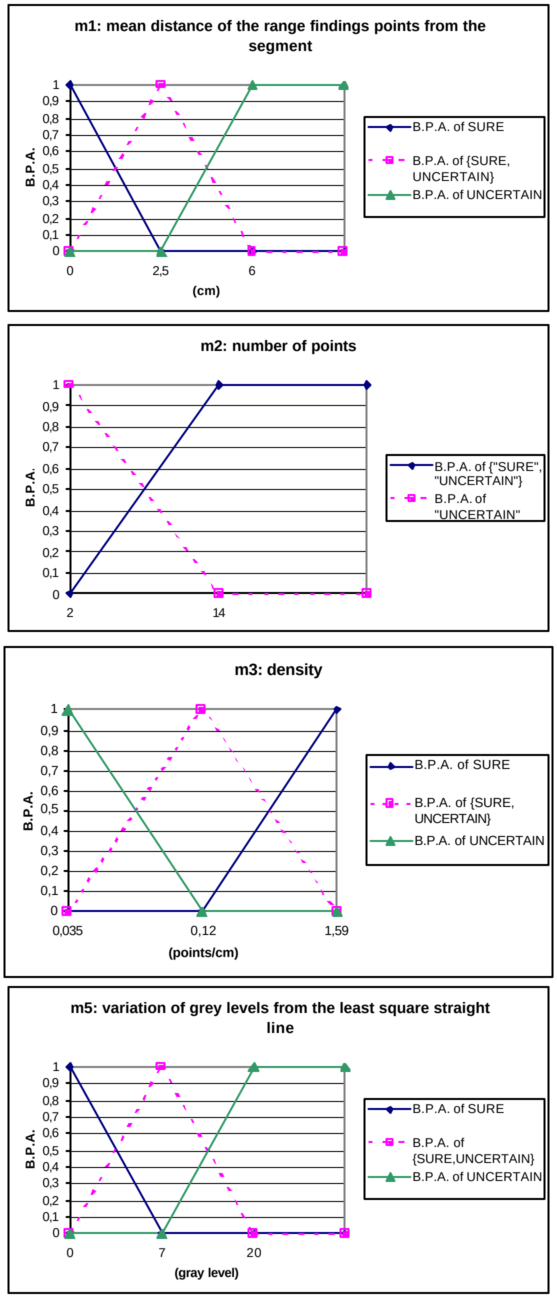


Figure 10: the B.P.As of the four classification criteria ({SURE,UNCERTAIN}=Q)

For the fourth criteria, the B.P.A. are:
weight 1:$m_4(\text{SURE})=m_4(\Theta)=0.6$, $m_4(\text{UNCERTAIN})=0.4$
weight 2:$m_4(\text{SURE})=0$, $m_4(\Theta)=1$, $m_4(\text{UNCERTAIN})=0$
weight 3:$m_4(\text{SURE})=0.3$,$m_4(\Theta)=0.7$,$m_4(\text{UNCERTAIN})=0$
weight 4:$m_4(\text{SURE})=0.6$,$m_4(\Theta)=0.4$,$m_4(\text{UNCERTAIN})=0$
weight 5:$m_4(\text{SURE})=1$, $m_4(\Theta)=m_4(\text{UNCERTAIN})=0$

We can then perform the combination calculation thanks to the Dempster-Shafer rules [8][11]. If the conflict coefficient k between the elements of the frame of discernment is superior to 0.7, it means that our criteria disagree. In this case, we decided that our segment is uncertain. If $k<0.7$, we compute the combination of belief functions for each element of the frame of discernment and we choose the class which has the maximal B.P.A.

The last stage (P3 on Figure 6) consists in eliminating the non significant segments in the final cooperative sensorial model. A non significant segment is characterized by a number of range finding points equal to 0 and a length (Cartesian distance) inferior to a predetermined threshold.

This stage permits to decrease the combinatory aspect of the matching stage and to increase the robustness.

This building algorithm enables to get a sensorial model where the number of exploitable primitives is more important than the number of primitives got by each sensor when it works individually. Besides, we obtain a certainty information of a segment by considering five criteria. This information will be used in the matching phase.

4 ABSOLUTE LOCALIZATION METHOD

The robot configuration is determined by matching the sensorial model, got by multisensor cooperation, with a theoretical map of the environment. The primitives used for this matching phase are segments. Therefore, all environment's elements like doors, walls, windows, radiators… are indexed as segments in the theoretical map.

For each segment, we have considered three correspondence tests, which are similar to these used by Crowley [7]:

❑ the angular difference between the two segments,
❑ the difference in length between the two segments,
❑ the distance between the centers of the two segments.

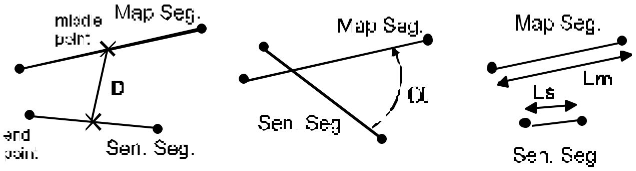


Figure 11: The three matching criteria.

The fusion of these three treatments is made thanks to the combination rules of the Dempster-Shafer theory [8]. Our frame of discernment is composed of two elements: YES and NO corresponding to those assertions : "Yes, we can match the two segments" and "No, we can not match the two segments". For each criterion, we have determined the Basic Probability Assignments (B.P.A.) m_1, m_2 , m_3 shown Figure 12.

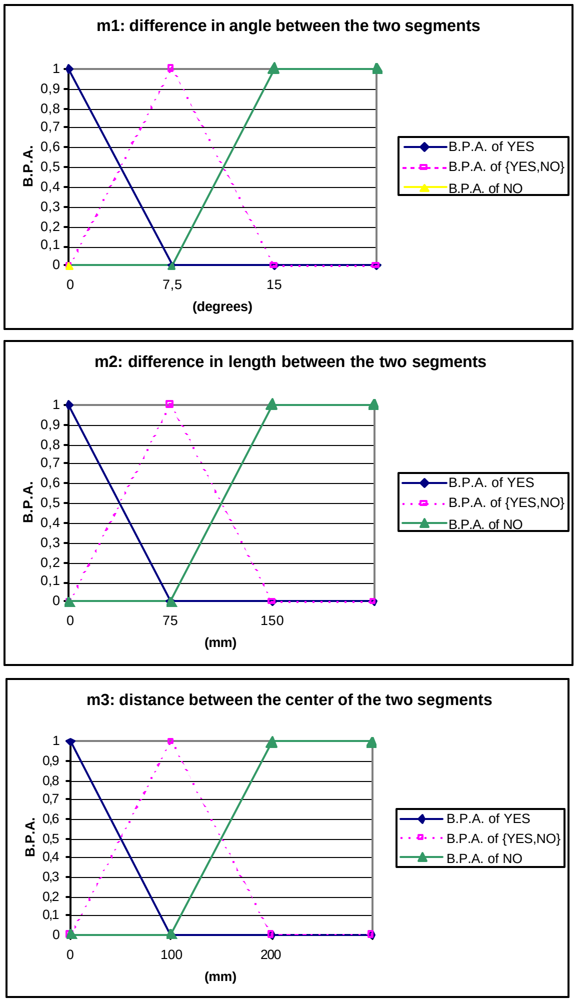


Figure 12: basic probability assignments of matching criteria ({YES,NO}=Q)

We can then perform the combination calculation thanks to the Dempster-Shafer rules [8]. Since we have three criteria, we first fuse the two first criteria.

The conflict coefficient between these two first criteria is:

$$k_{12} = m_1(YES).m_2(NO) + m_1(NO).m_2(YES) \qquad (1)$$

If $k_{12}<1$, the conflict is not complete and the combination of belief functions m_{12} for these two elements of the frame of discernment is given by:

$$m_{12}(YES) = \frac{m_1(YES).m_2(YES) + m_1(YES).m_2(\Theta) + m_1(\Theta).m_2(YES)}{1-k_{12}}$$

$$m_{12}(NO) = \frac{m_1(NO).m_2(NO) + m_1(NO).m_2(\Theta) + m_1(\Theta).m_2(NO)}{1-k_{12}}$$

$$m_{12}(\Theta) = \frac{m_1(\Theta).m_2(\Theta)}{1-k_{12}} \qquad (2)$$

Then we fuse the last criterion. We compute the conflict coefficient k (3) between this criterion and the two criteria we have fused above:

$$k = m_{12}(YES).m_3(NO) + m_{12}(NO).m_2(YES) \qquad (3)$$

If $k>0.7$, we think that the conflict is too high. So we decide to take a prudent decision: we don't match the two segments. If $k<0.7$, we compute the combination of belief functions for each focal element:

$$m(YES) = \frac{m_{12}(YES).m_3(YES) + m_{12}(YES).m_3(\Theta) + m_{12}(\Theta).m_3(YES)}{1-k}$$

$$m(NO) = \frac{m_{12}(NO).m_3(NO) + m_{12}(NO).m_3(\Theta) + m_{12}(\Theta).m_3(NO)}{1-k}$$

$$m(\Theta) = \frac{m_{12}(\Theta).m_3(b)}{1-k} = \frac{m_1(\Theta).m_2(\Theta).m_3(\Theta)}{1-k} \qquad (4)$$

The segments are matched if B.P.A. for the *YES m(YES)* is superior to the B.P.A. for the *NO m(NO)*.

The first stage of this localization algorithm consists in determining a list of sensorial segments *Ls* which have a strong probability of existence. This segments are the "SURE" segments obtained during the fusion stage.

We consider that the length of these segments has been determined with a good accuracy. So, our starting correspondence test is the length of a segment.

In the second stage, we consider a segment Ls_k from the list *Ls* and we search the theoretical map segments which length is similar to the Ls_k segment length. Each found theoretical segment is superposed on the sensorial segment Ls_k and we apply the third step in order to test the correspondence of the other sensorial segments.

The third step consists in applying the three criteria describe above on all the segments on the list *Ls* except the segment Ls_k. A segment is matched if the B.P.A. for the *YES* is superior to the B.P.A. for the *NO*. To choice the optimal matching solution we calculate a *V* criteria. For each matched segment pair, we increment this *V* coefficient which characterizes the robustness of the global matching. *V* is managed with the following algorithm:

```
Given:
-   B the B.P.A. for the YES of the matched
    segment pair
-   W a weight linked to the segment's class
    (SURE segment: w=3, UNCERTAIN segment:
    w=1).
FOR each global matching
      V=0
      FOR each segment matched
             V = V + (B*W)
      END
END
```

So we can see that *V* is an interesting and discriminative indicator of the global matching relevance since *V* takes into account the class of each matched segment ("SURE", "UNCERTAIN") and the quality of each matched pair (through the B.P.A. for the *YES*).

These three steps are then repeated for all the *Ls* list segments. The final solution is the one which permits the maximal *V*.

5 Experimental results

To test the robustness of our localization algorithm, we have performed it on several sensorial acquisitions made in an indoor environment (Figure 13). The two omnidirectional acquisitions are made when the robot is stopped. The omnidirectional acquisitions and the localization algorithm are computed in a Pentium PC located on our mobile robot. A Matrox Meteor II video card is used to acquire the omnidirectionnal image and the laser acquisition. Our experimental perception system is shown on Figure 13.

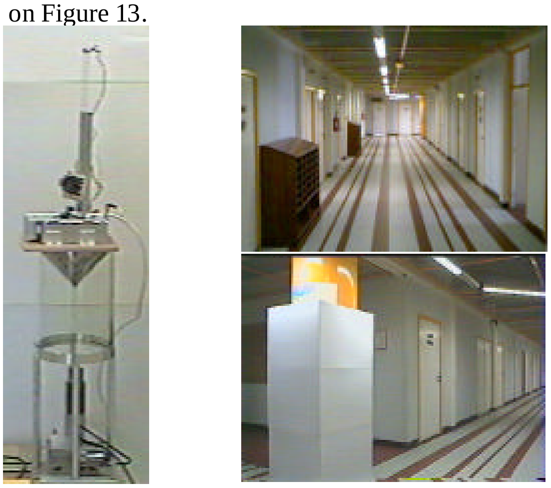
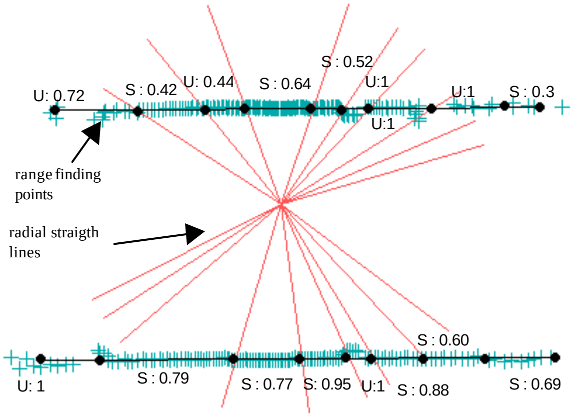


Figure 13: Our global omnidirectional perception system and the experimental indoor environment.

In order to show the interest of our cooperative approach, we have tested our localization method on symmetric environments (the first picture on Figure 13). The use of one sensor individually instead of the two sensors emphasizes the robustness problem: a strong failure rate has been observed for the matching phase when we use only one sensor [9].

The first environment is a long corridor (length: 50 meters). Figure 14 shows a sensorial model got with our cooperative approach. The robot is located in the middle of the corridor (Figure 13). We can see on Figure 14 the final decomposition on an set of segments which represent doors and parts of wall. We show on this figure the radial straight lines obtained with the omnidirectional conic mirror. We must note that, for this environment, the depth sensor would not have been able to localize the robot: two parallel identical segments would have been detected. The SYCLOP system used alone would have posed the problem of environment symmetry. We can also remark that uncertain segments are the segments which are far from the robot (not well aligned) or which correspond to the pillars of the corridor (not detected during the Duda-Hart segmentation stage). The robot final position successfully obtained shows the robustness of our method and its accuracy. We have indeed a position error of 8cm and an orientation error of 3 degree.



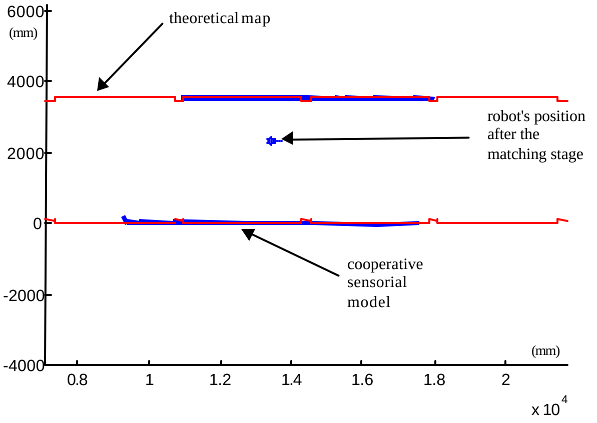


Figure 14: the cooperative sensorial model with the segments classification and the BPA(U=UNCERTAIN, S=SURE) (first figure) and the final position determination corresponding to the optimal matching

We show on Figure 15 results obtained in an other symmetric environment: a laboratory square hall.

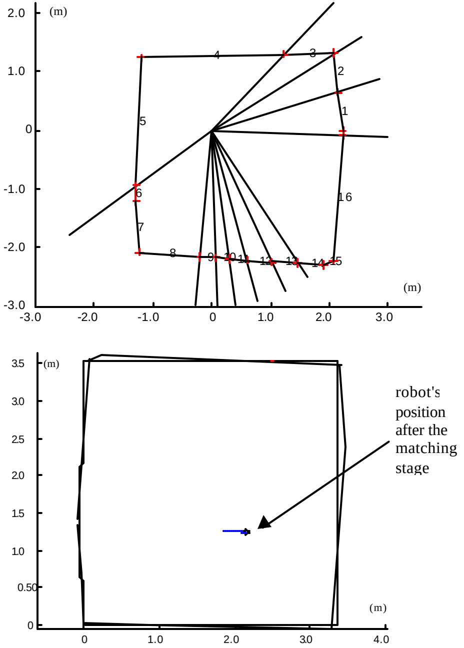


Figure 15: cooperative sensorial model and final position determination in a hall environment.

The same remarks can be done: the use of the two sensors provides enough sensorial information to enable the matching algorithm to converge to a coherent solution.

The third environment is the end of the corridor shown Figure 13. This environment constitutes a favorable experimental configuration: it is not symmetric and it has an important number of exploitable landmarks (figure 10). We can note here on several robot's configuration determination that our matching selection criteria is highly discriminative: the good configuration has been computed on all the acquisitions.

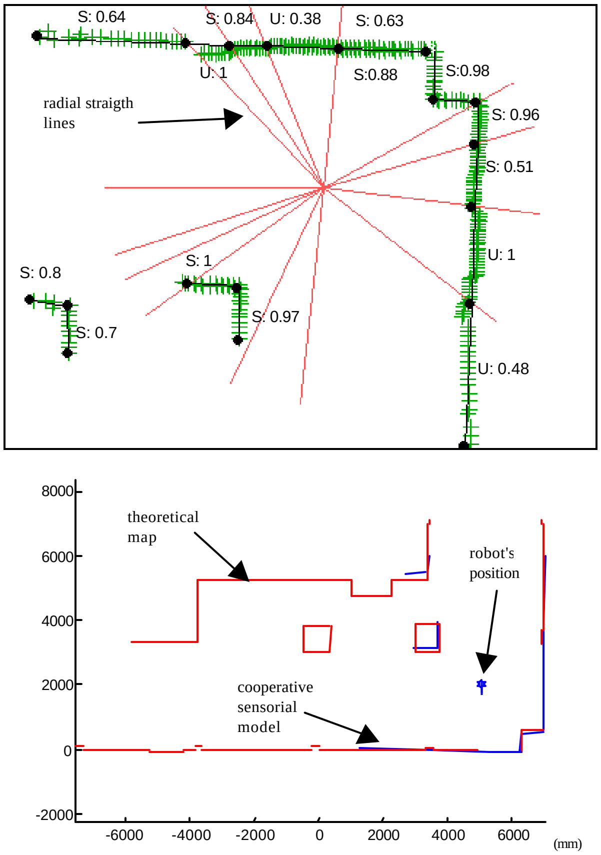


Figure 16: cooperative sensorial model (first figure) and final position determination.

Finally on a complete path makes in the corridor by our robot mobile SARAH, we could note on 40 acquisitions that, on the one hand, all the absolute configurations have been determined correctly, and, on the other hand, the mean error was equal to 11 cm in position and 3 degree in orientation.

In spite of an important combinatory aspect, our cooperative localization method proves to be robust and particularly accurate.

6 CONCLUSION

We have presented in this study an absolute localization approach based on the cooperation between two omnidirectional sensors: an omnidirectionnal vision sensor and a range finding sensor. This association allows to treat two types of complementary data. Then we obtain a highly descriptive sensorial model which integrates an important number of primitives and enables to increase the robustness of the matching stage. We classify also every sensed segment in two reliability classes according to five criteria fused thanks to the Dempster-Shafer rules. The absolute localization paradigm based on this matching stage takes into account several criteria which are merged with the Dempster Shafer rules. The choice of the optimal matching is based on a highly discriminative criteria which associates the segment reliability classes and a B.P.A. linked to the matching stage. We have tested our cooperative absolute localization algorithm on several particular environment like for example symmetrical environment. On the one hand, we can note on these experimental results that the robot's configuration determination is realized in a unique way and on the other hand the absolute robot's configuration is calculated with a relatively weak systematic error.

REFERENCES

[1] Y. Yagi, Y. Nishizawa, M. Yachida, "Map-based navigation for a mobile robot with omnidirectional image sensor COPIS", IEEE Trans. on Robotics and Automation Vol. 11, pp. 634-648, October 1995.

[2] J.Gomes-Mota, M.I. Ribeiro, "A multi-layer robot localisation solution using a laser scanner on reconstructed 3D models", Proc. on the 6th Int. Symposium on Intelligent Robotic Systems, Scotland, 1998.

[3] P. Bonnifait, G. Garcia, "Design and Experimental Validation of an Odometric and Goniometric Localization System for Outdoor Robot Vehicles.", IEEE Trans. on Rob. and Aut. Vol. 14, No 4, pp. 541-548, August 1998.

[4] C. Drocourt, L. Delahoche, C. Pegard, C. Cauchois , "Localization method based on omnidirectional stereoscopic vision and dead-reckoning", Proc. of the IEEE Int. Conf. on Int. Robots and Systems, Korea, October 1999

[5] H.R Beom, H.S. Cho, "Mobile robot localization using a single rotating sonar and two passive cylindrical beacons", Robotica, Vol. 13, pp. 243-252, 1995.

[6] J.A. Perez, J.A. Castellanos, J.M.M. Montiel, "Continuous localization: vision vs. laser", IEEE Proc. of Int. Conf. on Rob. and Aut. pp 2917-2923, Detroit, May 1999.

[7] J. Crowley, "Navigation for an intelligent mobile robot", IEEE Journal on Robotics and Automation, Vol. RA-1, n°1, pp. 31-41, March 1985.

[8] G.A. Shafer, "A mathematical theory of evidence", Princeton : university press, 1976.

[9] A. Clérentin, C. Pégard, C. Drocourt "Environment Exploration Using an Active Vision Sensor", Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS'99), Korea ,October 1999.

[10] J. Hollingum, "Caterpillar make the earth move : automatically", Industrial Robot, Vol. 18, N° 2, pp. 15-18, 1991.

[11] A. Dempster, "Upper and lower probabilities induced by a multivalued mapping", Annals of mathematical statistics 38:325-339, 1967.

Measuring Mobile Robot Performance: Approaches and Pitfalls

Gaurav S. Sukhatme
gaurav|@usc.edu
Robotics Research Laboratories
Department of Computer Science
Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, CA 90089-0781

Abstract

We consider the problem of measuring the performance of an intelligent mobile robot system. We believe that systems are intelligent because their capabilities are more than the sum of their parts. Therefore any piecemeal efforts to measure the performance of an intelligent system are bound to fail. Further, metrics of utility are more useful to designers than something as abstract as intelligence. We describe a task-based, multiple-criteria technique that combines two benchmarks to result in a metric for navigation. A case study of two robots is presented, which were evaluated and compared using the metric.

1 Introduction

We consider the problem of measuring the performance of an intelligent mobile robot system. We believe that systems are intelligent because their capabilities are more than the sum of their parts. Therefore any piecemeal efforts to measure the performance of an intelligent system are bound to fail. Only measuring performance along a single skill axis is also clearly limiting since intelligence does not boil down to a single skill or capability but rather arises due to a complex interplay between a multitude of capabilities. We strongly advocate the measurement of task-oriented quantities which establish the utility of a system. To this end, measuring performance along *several axes* is clearly important but brings with it several challenges:

- What should the axes be ?
- How do we ensure the axes span the space we want to benchmark ?
- Does an “orthogonal” set of axes exist ?
- How should the performance measures along these axes be combined ?

In this paper we describe a task-based, multiple-criteria technique that combines two benchmarks to result in a metric for navigation. A case study of two robots is presented, which were evaluated and compared using the metric.

2 Previous Work

Due to space limitations we limit ourselves to a brief survey of evaluation techniques for mobile robots. The so-called static evaluation techniques are specifically designed for measuring stability when the robot is stationary and when it is moving in a statically stable fashion. The primary method of choice is an energy based stability measure as an evaluation function. In work by Nagy et al. [7] two modes of walker stability are characterized namely stance stability and walker stability. Both use the amount of energy needed to destabilize the walking robot as a measure of the stability of the robot. The stance stability is identical to the energy stability margin defined by Messuri et al. in [5] as the minimum work that must be done on a robot walker to tip it over an edge of a support boundary.

Early work on robot stability was due to McGhee et al. [4] who defined the support polygon as the convex hull of the projections of all contacting points on a horizontal plane. In [3] the authors define a conservative support polygon with the motivation that the walking robot should retain its stability in the event of a single leg failure. Of the above energy based measures of stability the work of Nagy et al. is the most general since it includes compliance of the mechanism and depends on the terrain that is underfoot.

In [1] the authors discuss several evaluation criteria for comparing three configurations for the design of a walking robot. Some of the evaluation criteria were foothold selection area, stride length, static stability and energy stability. The important tradeoff was stride vs. stability, based upon which the circulating configuration for Ambler was chosen.

Dynamic evaluation techniques are so named because they focus on properties related to motion. Wilcox [10] introduced a metric called the MCC (Mobility Characteristic Curve) to measure the ability of a robot to surmount obstacles. The obstacle was a cylinder of (theoretically) infinite length and diameter d which was buried to a depth $d/3$ in an inclined plane of slope s composed of loose sand. The MCC was defined as the plot with s on the horizontal axis and the diameter of the largest cylinder that the robot could surmount (in dimensionless units based upon its length) on the vertical axis. The proposed figure of merit was the area between the co–ordinate axes and the MCC. The two main achievements of this method were its independence of scale and easy reproducibility. Its chief drawback was that it used a simple obstacle geometry and did not evaluate the entire system in a mission oriented way.

Lietzau [2] proposed a set of benchmarks to assess the performance of a Mars microrover. These benchmarks were divided into five categories namely, mobility, navigation and control, science, autonomy and environmental. A set of weights was assigned to these categories based upon their importance by the system designers and mission specialists. The weighted sum of the individual benchmarks was then proposed as a figure–of–merit. Lietzau's work is a thorough description of the individual subsystem tests that are a necessary part of evaluation but does not focus on the system level evaluation that we emphasize here. Though it was never formally characterized as such, Lietzau's evaluation technique is an example of a Linear Programming approach to solve the problem of evaluation.

3 Case Study

The evaluation methodology that we propose here is for a particular robot mission - exploration of an unknown planetary surface. The area to be explored is assumed to contain rocks whose positions are not known *a priori* to the robot since it is presumed to be in unfamiliar surroundings. The robot mission is to perform scientific experimentation on rocks that are "interesting". We propose two evaluation functions in this study based on robot displacement as a function of mission time and energy consumption.

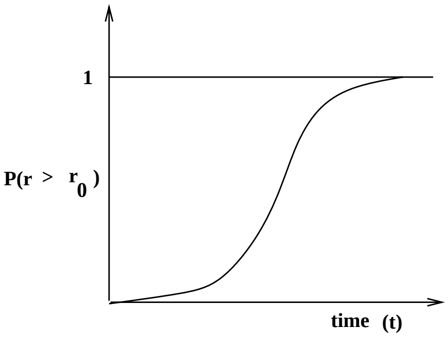


Figure 1: A Schematic of $P(r > r_0)$ vs. Time

3.1 The Cost Functions τ and η

The basic intuition behind the two cost functions proposed is to develop a nondimensional measure of the robot's ability to cover distance. The idea is to measure how "good" a particular robot design is by measuring how far the robot travels from the start location as a function of the time elapsed and the energy consumed by it. At first sight it may seem like the consumption of these two resources is extremely well correlated. This is indeed the case for straight-line travel on level ground with no obstacles. However, in the presence of obstacles it is not so - especially since the energy consumption of the system changes dramatically depending on whether it is at a standstill or in motion.

We define a trial as an autonomous traverse of the terrain by the robot in a particular instantiation of obstacle placement from start to goal. Using multiple trials we estimate the probability that the displacement $r > r_0$ for different values of the time t. A schematic of this probability as a function of time is shown in Figure 1. The main intuition is that the quicker this curve rises (close) to 1, the better the time utilization of the robot. Further, good time utilization also dictates that this curve be monotonic increasing. For the purposes of evaluation one is interested in the robot covering some displacement r_0 within some time t_0. In other words we expect some minimum performance for a limited resource (time).

The above requirement means that Robot A should be assigned a higher score than Robot B in Figure 2. This can be achieved by defining the area under the curve from $t = 0$ to $t = t_0$ as a metric. In order to compare robots of different size we measure displacement ($r = kl$) in terms of the number k of robot lengths l. We also measure time in nondimensional terms by

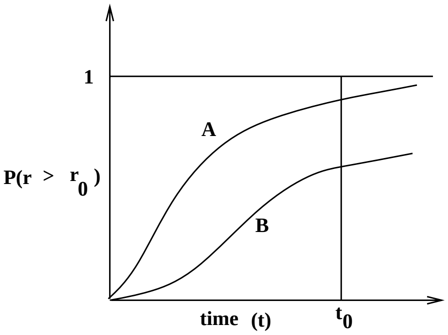


Figure 2: A Schematic Comparison of $P(r > r_0)$ vs. Time for Robots A and B

multiplying it with v/l where v is the robot velocity.

Let $\pi_{kl}(t)$ denote the probability of reaching a displacement kl as a function of time.

Definition 1 *The time figure of merit is defined as*

$$\tau = \int_0^{t_0} \pi_{kl}(t)dt \tag{1}$$

In a similar manner we plot the probability of reaching a displacement kl as a function of the energy e consumed. Energy is converted to a nondimensional quantity by dividing it by mgl where m is the mass of the robot and g is the acceleration due to gravity.

Let $\pi_{kl}(e)$ denote the probability of reaching a displacement kl as a function of energy.

Definition 2 *The energy figure of merit is defined as*

$$\eta = \int_0^{e_0} \pi_{kl}(e)de \tag{2}$$

Note than both figures of merit are non-dimensional.

3.2 The Robots: MENO and Marscar

MENO is a 12 DOF statically stable quadruped designed and constructed for this study in the USC Robotics laboratory. Each leg is a rotary-rotary-prismatic (RRP) design. The body of the robot and the first two links of each leg are in the horizontal plane and the prismatic joints (the most distal joint of each limb) are in the vertical plane. This orthogonal design was inspired by the design of Ambler [1].

The wheeled robot Marscar is 4 wheeled rover with Ackerman steering.[1]

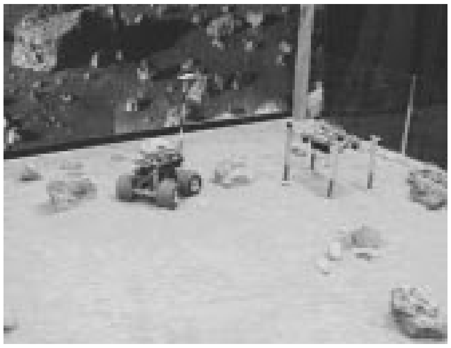


Figure 3: MENO and Marscar in a Simulated Martian Environment

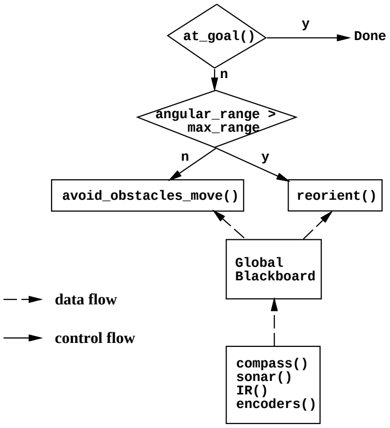


Figure 4: The Control Architecture for the Wheeled Robot

There are two main behaviors that drive both robots. They are `avoid_obstacles_move()` and `reorient_to_goal()`. A schematic of the control architecture is shown in Figure 4.

Onboard computing is all done on a custom board built around a Motorola 68332 microcontroller. A tether is used to supply offboard power for extended testing and for gathering telemetry. The testing is all done in a 3.5 m ×3.5 m sandbox. A single camera suspended 3 m above the center of the sandbox is used for tracking the robot's position. We do *not* use the overhead camera as a source of information for navigation; navigation is done by dead reckoning using information measured by onboard sensors only. The sand surface is nominally flat but not precisely so.

[1] Ackerman steering maintains a particular relationship between the steer angles of the inner and outer wheels in order that the entire robot turn about a single point.

```
Loop until at goal:
    If obstacle in front
        Compute 'good' detour direction
        Detour
    Else
        If goal within angular range limits
            Move forward
        Else
            Reorient towards goal
        Endif
    Endif
EndLoop
```

Figure 5: The Navigation Algorithm

3.3 The Navigation Algorithm

Both robots above use the same behavior-based navigation algorithm. There are two[2] basic behaviors; 1. Reorient towards goal and 2. Avoid obstacles. The basic idea is for the robot to keep track of its current position using knowledge of its kinematics and proprioceptive sensors (such as wheel encoders on Marscar and joint angle measurements on MENO). The estimator running on board the robot performs a simple dead-reckoning calculation to estimate position and orientation at every move. The 'avoid obstacles' behavior is also fairly simple - if an obstacle is seen the robot will attempt to detour around it (while keeping track of its position as mentioned above). If no obstacle is blocking the robot, it will attempt to move towards the goal, re-orienting itself if necessary. The navigation algorithm is reactive. A schematic outline of the algorithm is given in Figure 5.

An interesting part of the detour behavior is the use of global information. When an obstacle is detected the reactive strategy is to backup and turn. The direction of the turn is dependent on the current location of the robot and the commanded goal location in global coordinates. The turn direction that reduces the difference between the robot angle and the desired goal angle θ_g is chosen and executed. A purely local strategy would pick one direction at random but the reactive obstacle avoidance behavior is modified to use some global information viz. the goal position.

We also adapt the angular range during a traverse. The basic observation is that small angular errors when the robot is far away from the goal lead to large position errors later. To avoid this we keep the angular range limits (within which no reorientation is necessary) small when the robot is far away from the goal. These limits are progressively increased as the robot nears the goal.

The experiments were performed in a simulated Mars terrain comprised of a crushed red brick sand mixture. The mixture was spread evenly in a 3.5 m by 3.5 m sandbox to a depth of 0.25 m. The sandbox was populated with rocks of varying size (between 0.04 m and 0.2 m in diameter) to simulate Martian rock distributions. The density of the rocks was equal to the Mars nominal density from the Moore distribution [6].

Since the evaluation functions use probability estimates from numerous mission trials, the experimental protocol consists of many robot traverses from start to goal locations in different instantiations of Mars nominal terrain. There are three main loops. During a particular instantiation a number of trials are performed with different start and goal locations. During the course of each of these trials (as the robot is navigating from start to goal) the offboard computer is monitoring time. When a certain time interval δ_t is reached the overhead vision system images the robot and the image is stored with a timestamp. When the current trial is over the sequence of images taken is postprocessed to extract the (x, y) location of the robot as a function of elapsed time. This information is stored in a file and the next trial begins. The procedure is terminated when all the exemplar start/goal locations have been used in every exemplar terrain. The protocol for energy is exactly the same as the time trials but instead of monitoring the time elapsed, the power draw is monitored. Using this a running total of the energy consumed is maintained. When the energy consumption reaches a threshold δ_e the robot is imaged.

Once the data recording the position and orientation of the robot is obtained using the protocol described above, it is processed to create plots of the required probability estimates that yield the previously defined figures of merit that we are interested in. The data processing steps for the time trials are as follows:

- Fix a given time resource value (t_0)
- Fix a required minimum displacement (r_0)
- Build a plot of $\pi(r > r_0)$ vs. t
  1. for each of the n data sets, $\forall t < t_0$ compute $r = \sqrt{(x - x_s)^2 + (y - y_s)^2}$
  2. a = number of r values greater or equal to r_0
  3. use a/n as the required probability estimate
- Compute nondimensionalized $\tau = \int_0^{t_0} \pi_{r_0}(t)dt$

[2] The legged robot also has balancing and gaiting behaviors at a lower level. They are discussed elsewhere [8]

- Repeat above steps for different values of r_0 and t_0

The data processing steps for the energy trials are similar. In both outlines above (x_s, y_s) is the robot start location and $\pi(r > r_0)$ denotes the probability that the displacement r from the start location is greater than r_0.

4 Data Analysis

The experiments were performed in simulation and with the physical robots. The datasets discussed here thus contain results from both. We will however restrict ourselves to a discussion of the datasets from the physical robots since space constraints do not allow a complete discussion here. The interested reader is referred to [9] for a complete account.

4.1 Mobility Trials and Clustering in Tradeoff Space

The first step in calculating the figures of merit is to calculate the probability of reaching k robot lengths as functions of time and energy. Since we have multiple trials we estimate this probability as the fraction of trials in which the displacement was greater than kl as functions of time and energy consumption. In Figure 6 the probability of Marscar reaching the threshold displacement kl is shown for various values of k. The quantity l is intended to be a measure (with dimensions of length) of the robot size. We use the cube root of the volume of the smallest rectangular box in which the robot can be packed. For Marscar $l = 0.35$ m. All the trials were done in Mars nominal distributions. One can see a reasonable agreement between the simulated dataset and the dataset collected from the physical robot. The simulated dataset consisted of 200 trials and the physical dataset consisted of 40 trials. The probability estimates of the simulated dataset are smoother compared to the physical dataset due to the larger sample size. The general behavior of the family of curves shown in Figure 6 is a monotonic rise to saturation. The interpretation of these curves is the likelihood of success (at navigating through the obstacle field) as a function of the available resource (time). A higher k value corresponds to a longer traverse and thus involves greater ability in penetrating obstacle fields. As k is increased for the same robot the probability of achieving the same degree of success decreases.

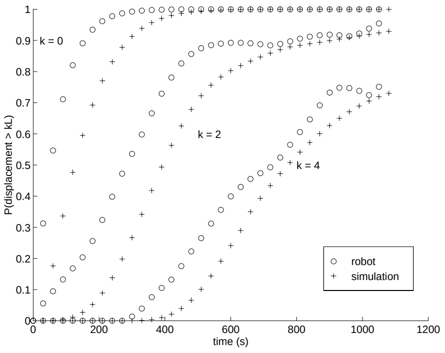


Figure 6: Marscar - Probability of reaching threshold displacements vs. time in Mars nominal terrain

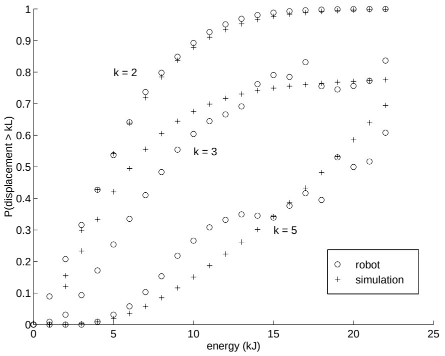


Figure 7: Marscar - Probability of reaching threshold displacements vs. energy in Mars nominal terrain

In Figure 7 a family of curves is shown which plot the probability of Marscar achieving a threshold displacement kl as a function of energy consumed. As in the case of the plots in the previous figure, the probability of greater success shows an asymptotic rise to saturation. Figure 7 shows the probability estimates for the simulated as well as physical datasets. As one can see there is a good match between the two. As in the previous case larger k values imply longer missions and thus are harder to achieve for the same value of the energy resource. Performance degrades as k is increased. As in the time trials with Marscar, the physical datasets in Figure 7 are the result of 40 trials and the simulated datasets are the result of 200 trials.

In order to compute the figures of merit for MENO in Mars nominal terrain we follow the same data analysis procedure as before. The curves showing the plots of the probabilities of achieving the threshold displacement kl as a function of time elapsed are shown in Figure 8. As in the previous cases increasing values of k signify longer missions. For MENO $l = 0.47$ m. The

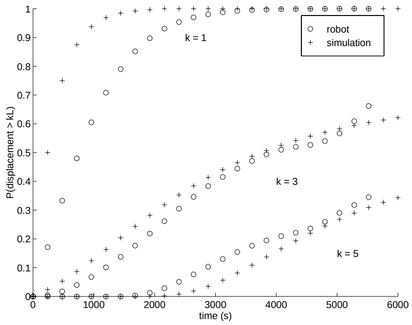


Figure 8: MENO - Probability of reaching threshold displacements vs. time in Mars nominal terrain

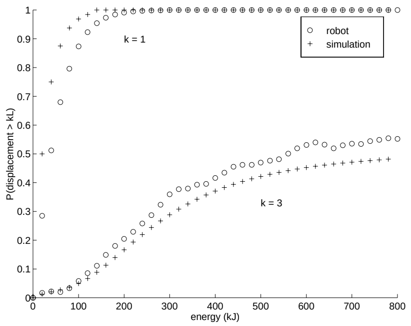


Figure 9: MENO - Probability of reaching threshold displacements vs. energy in Mars nominal terrain

physical datasets shown in Figure 8 were computed using 40 trials in Mars nominal terrain and the simulated datasets were generated using 200 trials in simulation.

The last datasets of interest in the current series are the behavior of MENO as a function of the energy consumed in Mars nominal terrain. The relevant plots are shown in Figure 9.

In the notation of Chapter 4 we now have plots of $\pi_{kl}(t)$ and $\pi_{kl}(e)$; the probabilities of the achieving certain threshold displacements as functions of time and energy. Using $t_0 = 40$ min and $e_0 = 200$ kJ as representative numbers for the mission under study we calculate the two figures of merit using Equations 1 and 2 for different values of k. These values are shown in Table 1.

Figure 10 shows the τ and η values for the two robots in the tradeoff space. The lower left hand side of the plot (signifying lower evaluation scores) is the space occupied by the legged robot. The wheeled system has better scores on both time and energy axes. The evaluation functions are evaluated for 4 different values of k. Irrespective of the k value the wheeled robot outperforms the legged robot. The functions η and τ thus partition the design space.

Table 1: The Figures of Merit for MENO and Marscar for Different Traverse Lengths

k	Marscar		MENO	
	τ	η	τ	η
2	14.7	11504	2.7	6911
4	12.1	11429	2.1	5957
6	9.4	9718	1.1	3609
8	7.3	9635	0.8	2745

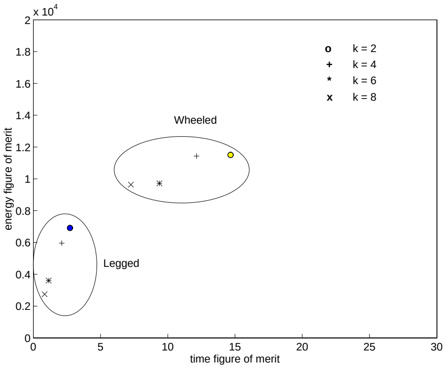


Figure 10: A Comparison of MENO and Marscar in Mars Nominal terrain for Different values of k

To illustrate the cause of the difference in the evaluation scores it is useful to re-examine Figures 6 and 7 on the same scale. This is done in Figure 11 where we show the probability estimates for both MENO and Marscar with $k = 5$ as a function of the time elapsed. Seen on the same axis it is obvious that the wheeled system does better with the 'area under the curve' metric since it is a lot faster than the legged system in this terrain (the Mars nominal rock distribution).

If Figures 8 and 9 are plotted on the same axis a similar conclusion can be drawn regarding the energy

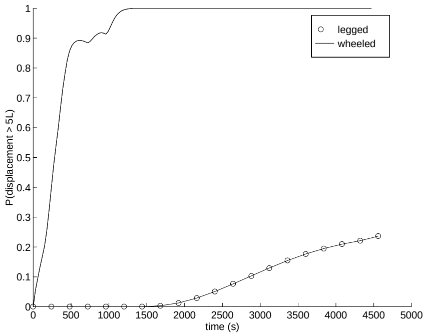


Figure 11: A Comparison of MENO and Marscar in Mars Nominal terrain for $k = 5$ as a Function of Time Elapsed

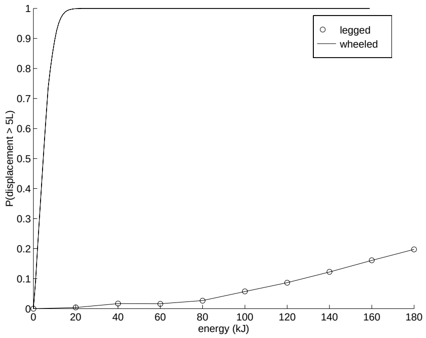


Figure 12: A Comparison of MENO and Marscar in Mars Nominal terrain for $k = 5$ as a Function of Energy Consumed

scores. This is shown (again for $k = 5$) in Figure 12. The wheeled robot needs far less energy to cover the same distance compared to the energy consumption of the legged robot over a similar distance for this particular rock distribution.

4.2 Sensitivity Studies - Environment

One of the objectives of this study was to measure the effects of changes in environmental parameters on the mobility metrics. The environment model used in this study is the distribution of rocks called the Moore distribution. In the vicinity of a previous mission to Mars (the Viking II mission) the density of rocks is much higher than the Mars nominal distribution used thus far. The effect of terrain clutter is very clearly seen in the two metrics. In the case of both robots, increased clutter leads to performance degradation. However it is interesting to note that the wheeled system is affected far more than the legged system. This is largely due to the fact that the increased clutter leads to significantly longer paths for the wheeled system whereas the legged system is able to go over many more obstacles and even though it is slower its performance is comparable to the legged robot. This is shown in Figure 13.

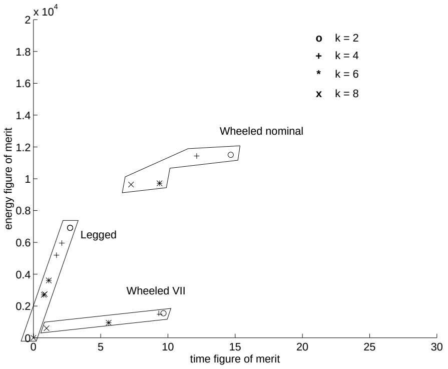


Figure 13: MENO and Marscar mobility in Viking II (cluttered) terrain for different values of k

As one can see in Figure 13 the Marscar cluster moves dramatically to the left and down when the terrain was changed from Mars nominal to Viking II. MENO performance also suffered as seen in Figure 13 but not as dramatically. For this environment, its energy figure of merit is better than Marscar.

4.3 Scalarization of the Metrics

The metrics τ and η can be combined into a single scalar metric using a weighted linear combination. From the data presented in this Chapter we see that the wheeled robot outperforms the legged vehicle along both dimensions in Mars nominal terrain. The scalarization chosen should preserve this ordering. A standard technique is to use a weighting function which is either linear or quadratic and maximize the combination of the two metrics. However the problem of how to choose the weights still remains. Instead of an ad hoc solution we use domain knowledge to postulate a feasible scalarization technique.

On one axis (τ) we are measuring the robot's ability to use time effectively and on the other (η) we measure effective energy utilization. The fundamental unit of conversion between them is the maximum power delivered by the onboard power source. If the power source is capable of delivering α W then we weight energy and time in the ratio $1 : \alpha$.

We computed the scalarized scores for $k = 6$ for the different cases reported in this Chapter using $\alpha_1 = 30$, $\alpha_2 = 40$ and $\alpha_3 = 50$. Using this scalarization technique it is clearer that in sparse obstacle distributions the legged system should be the preferred design while in dense obstacle distributions (such as the Viking II site) the nominal configuration of the legged robot MENO is the better design using these metrics and this particular linear scalarization.

5 Discussion

Values of the two metrics, τ and η for Marscar are significantly superior to the MENO values. The effect of obstacle clutter, though, is more pronounced on the wheeled robot.

There are three interesting aspects of the data presented here which form the basis for substantial future research. The first deals with the following design question: "In what parts of the design space are good designs found ?". At first glance it may seem like the answer is obvious - by definition it would seem like the designs leading to the highest values of the evaluation functions are the good parts of the design space. However, a closer look suggests that the real 'sweet spots' in the design space are those where the design is insensitive to changes in the environment. For example, MENO in its nominal configuration is insensitive to changes in rock density. If there is large variability in the expected terrain density it may be a better decision to pick a design like MENO even though it has low evaluation scores compared to other designs. We are thus led to believe that future scalarization efforts should include weighted contributions from select components of the evaluation gradient in addition to the values of the evaluation functions themselves.

The second interesting point also concerns the evaluation gradient. Locations in the design space where the evaluation gradient becomes very large also provide interesting insight into design methodology. We suggest that these locations in the design space signal a 'breakdown' in the current kinematic design and a discrete jump to a new structure is indicated (with higher articulation perhaps or with a larger number of wheels).

A third application of the metrics proposed here is to global optimization. While the technique for extrapolating performance shown here is local, it is possible to extend it by instantiating a chain of local models and following the evaluation gradient to an optimal set of parameter values.

6 Acknowledgments

The author gratefully acknowledges help from George Bekey in the development of these ideas. The author is partially supported by two DARPA programs; the Tactical Mobile Robotics (TMR) program under contracts F04701-97-C-0021 and DAAE07-98-C-L028, and the Mobile Autonomous Robot Software (MARS) program under grant DABT63-99-1-0015. The case studies described here were performed under the auspices of NASA grant #959816 from Caltech/JPL.

References

[1] J. Bares and W. Whittaker. Configuration of autonomous walkers for extreme terrain. *International Journal of Robotics Research*, 12(6):535–559, 1993.

[2] K. R. Lietzau. Mars micro rover performance measurement and testing. Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, December 1993.

[3] S. Mahalingam and W. L. Whittaker. Terrain adaptive gaits for walkers with completely overlapping leg workspaces. In *Proc. Robots 13*, May 1989.

[4] R. B. McGhee and A. A. Frank. On the stability properties of quadruped creeping gaits. *Mathematical Biosciences*, 3(3/4):331–351, 1968.

[5] D. A. Messuri and C. A. Klein. Automatic body regulation for maintaining stability of a legged vehicle during rough terrain locomotion. *IEEE Journal of Robotics and Automation*, RA-1(3):132–141, 1985.

[6] H. J. Moore and B. M. Jakosky. Viking landing sites, remote-sensing observations and physical properties of martian surface materials. *Icarus*, 81:164–184, 1989.

[7] P. Nagy, S. Desa, and W. L. Whittaker. Energy-based stability measures for reliable locomotion of statically stable walkers: Theory and application. *The International Journal of Robotics Research*, 13(3):272–287, June 1994.

[8] G. S. Sukhatme. The design and control of a prototype quadruped microrover. *Autonomous Robots*, 4(2):211–220, April 1997.

[9] G. S. Sukhatme. *On the Evaluation of Autonomous Mobile Robots*. PhD thesis, University of Southern California, May 1997.

[10] B. Wilcox. Mobility characteristic curve. Jet Propulsion Labs, IOM 3472-91-019, 1991.

Search Graph Formation for Minimizing the Complexity of Planning

Alberto Lacaze

Computational Intelligence Laboratory, ECE
University of Maryland, College Park

Stephen Balakirsky

Intelligent Systems Division
National Institute of Standards and Technology

Abstract

A large number of path planning problems are solved by the use of graph based search algorithms. There are a variety of techniques available to optimize the search within these graphs as well as thorough studies of the complexity involved in searching through them. However, little effort has been dedicated to constructing the graphs so that the results of searching will be optimized.

The commonly used approach for the evaluation of complexity assumes that the complexity of a path planner can be evaluated by the number of nodes in the graph. However, in many path planning problems (especially in complex, dynamic environments) the evaluation of the cost of traversing edges is the major culprit of computational complexity. In this paper we will assume that the complexity associated with the computation of cost of traversing an edge is significantly larger than the overhead of searching through the graph. This assumption creates non-trivial complexity results that allows to optimize the creation of the graph based on the computational power available.

We will present a numerical evaluation of several graph creation algorithms including the commonly used four and eight connected grid. Different scenarios for which ground truth is available are explored. Comparison among the graph creation algorithms reveals serious downfalls that are common practice throughout the literature.

1 Introduction

Planning can be defined as the process of finding the steps necessary to bring a system from an initial (current) state to a final (desired) state. Most planning techniques represent the planning problem in a graph $G(V, E)$. Where V is a set of vertices, and E is a binary relation on V [6, 7, 9]. The elements of the set V are called vertices and represent states. The elements of the set E are called edges and represent the ability of the system to move from one state to another. In planning graphs, the edges are ordered or unordered pairs of vertices, (v_i, v_j) where $v_i \in V$ and $v_j \in V$. A walk is an alternating sequence of vertices and edges, a trail is a walk with distinct edges, and a path is a trail with distinct vertices.

When solving a planning problem, we must find a path or plan from a starting vertex v_s to an ending vertex v_e while minimizing a cost function $C = \sum_s^e w_{ij}$ where w_{ij} is the cost of traversing the edge (v_i, v_j). Some planning problems can be solved by algorithms with polynomial complexity. Unfortunately, these tractable set of problems covers only a few of the relevant problems encountered in path planning. Most problems, however, can only be solved by polynomial algorithms on non deterministic machines, ie NP. For a thorough study on the problem of tractability and its taxonomy see [8].

One very useful tool when fighting the computational complexity of planning is the creation of hierarchies of planners. The Real-time Control System (RCS) reference model architecture is one such architecture and it has been successfully applied to multiple diverse systems [1, 3]. The target systems for RCS are in general, complex control problems. Although it has been shown [2, 10] that the complexity of a control problem is reduced by the use of a hierarchical control system, the reduction of error as a function of complexity at one level of the hierarchy has been mostly overlooked.

The complexity of search algorithms inside a graph has been thoroughly studied [11, 13, 14]. However, with few exceptions [4, 12], little attention has been paid on how the graph should be built with some exceptions [4, 12]. In most cases, it is recommended that the graph for search on "empty space" should be built using grids, Voronoi diagrams, or visibility graphs. It

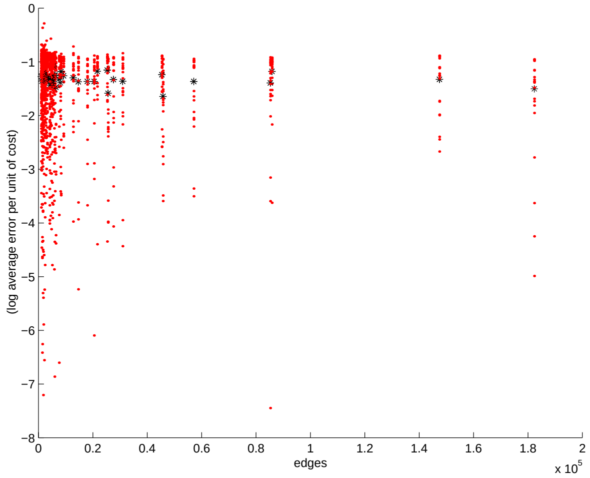


Figure 1: Average error for a 4 connected grid.

is not clear from the literature which of these methods should be used and when. Moreover, in most cases the complexity of algorithms is calculated solely based on the number of vertices in the graph. In most path planning problems, the computational complexity of calculating the cost of the edges is orders of magnitude higher than the actual time spent searching through the graph once these values have been calculated.

2 Numerical Exploration of Graph Creation

In order to compare the different graph formation algorithms, we started by defining a simple test scenario. The analytical closed form evaluation of the complexity of finding the optimum path taking under consideration the placement of the vertices in the solution space becomes easily intractable. Therefore, we decided to study the problem numerically. In the experiments presented in this paper, simple Euclidean distances were used to calculate the cost of traversing the edges. The advantage of using this measure is that we have ground truth. We assumed that the Euclidean distance is calculated with an accuracy of five significant figures.

2.1 Grid Based Graphs

By far, the most commonly used graph for search in planning algorithms is the four-connected square grid. In this kind of graph, the vertices are placed at regular intervals and it is assumed that each vertex is connected to four (or eight) of its closest neighbors.

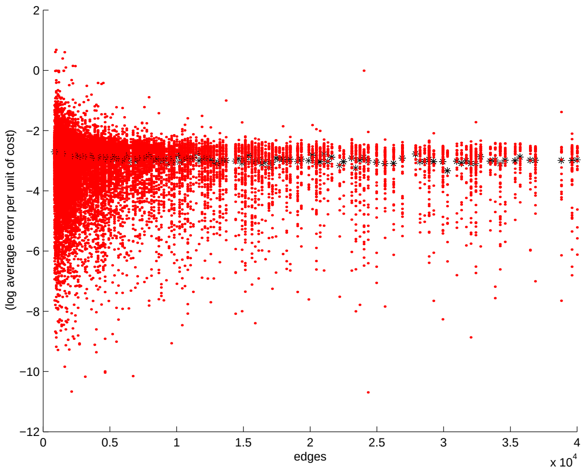


Figure 2: Average error for a 8 connected grid.

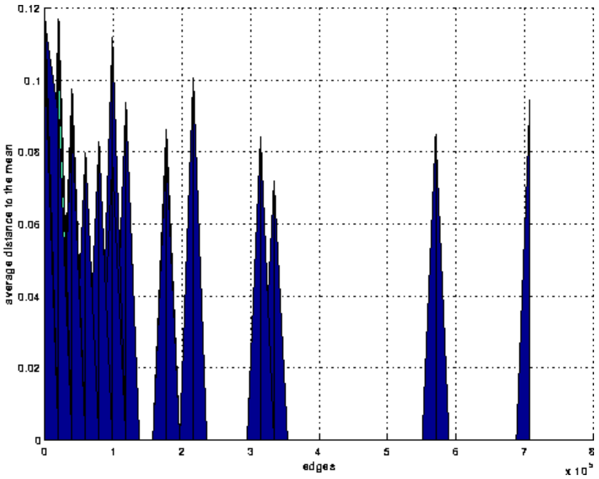


Figure 3: Average distance to the mean.

We built a two dimensional four-connected square grid with a random number of vertices. We repeated this experiment several times. Figure 1 shows $log(error)$ where $error$ is defined as

$$error = abs(d_{s,e} - ((\sum_{vs}^{ve} d_{i,i+1}) + d_{s,v_s} + d_{e,v_e})) \quad (1)$$

s is a randomly selected starting point, e is a randomly selected ending point, v_e is the closest vertex in the graph to e, v_s is the closest vertex in the graph to s, $d(i,j)$ is the Euclidean distance between two points. Please note that this cost function may underestimate the real error of traversing the planned graph as it is assuming that d_{s,v_s} and d_{e,v_e} are Euclidean. This is a best case scenario.

The summation in the equation represents the added cost of the optimal path through the graph. The average error (marked with a black star in the Figure) is kept constant as the number of edges is changed. The different values at a particular number of edges correspond to the different number of times that the experiment was performed using different e and s.

Figure 2 shows the error function shown in 1 applied to a eight-connected grid. As expected, the error function settles at a lower error. By comparing the 4-connected grid to the 8-connected grid we can appreciate that the average error decreases with the higher connectivity, however in both cases, the error quickly settles to a constant value.

Please note that in both cases, increasing the number of edges, and therefore increasing the computational complexity gives us very modest improvements of the final cost. Another problem found experimentally with the 4 and 8 connected grids using this cost function is that there are many paths that have exactly the optimal cost. This has the effect that the optimal path that the algorithm will choose, may wander off the "expected" straight path line from e to s. In other words, many paths within the parallelogram defined by v_s and v_e have exactly the same "optimal" cost. Another effect that results from square grids is that the error varies significantly depending on the direction of travel. A numerical evaluation of this deviation can be appreciated by examining Figure 3. The large average distance to the mean is due to the fact that some s and e happened to be horizontal or vertical, therefore giving small error, while some created a very costly stair-step paths through the graph.

2.2 Shaking the Grid

Some of the pitfalls of the grid based graphs can be avoided by:

1. Shaking the vertices within the grid. In other words, building a square grid, adding a random displacement to the vertices, and finally connecting all the vertices that are within a neighborhood. The size of the neighborhood dictates the vertices to edges ratio. This has two effects:
   (a) Break the ties among optimal paths so that only one path is found to be optimal. This is very helpful in re-planning systems as it forces to commit instead of randomly flipping among the set of "optimal" paths.
   (b) Create a more uniformly distributed set of vertices where all " directionalities" are represented.
2. Create higher connectivity rates (higher than in the 8-connected grid).

Figure 4 through Figure 7 shows the results of a set of experiments run using the above principles. To compute these figures, the vertices of the grid are placed first in a grid pattern where each point is l apart from its closest neighbor. Next, a random vector is added to each vertex of maximum amplitude $3l$. All vertices within a distance threshold are then connected. By varying the connection threshold, different ratios between the number of nodes and the number of edges are achieved. We can see from Figure 4 that the error decreases as the number of edges increases, approaching the 10e-5 mark set by the 5 significant figures used to calculate the Euclidean distances. Figure 5 shows a top view of the same numerically found error. We can see that even a simple Euclidean cost function creates ripple effects in the final cost.

If we take the assumption that the computational complexity is directly proportional to the number of edges (as it is in most cases), we can see in Figure 8 the error function as a function of the number of nodes. The almost counter-intuitive results can be explained from the fact that by increasing the number of vertices the average cost of an edge decreases. In Figure 9 we assumed that we could only calculate the cost of 40000 edges. By visual inspection of Figure 9 we can determine that the least error is given by about 2000 vertices, and therefore creating a graph where each vertex has 20 connected neighbors.

3 Vehicle Planner Example

In order to validate the above rules of thumb, several experiments were conducted using the Demo III

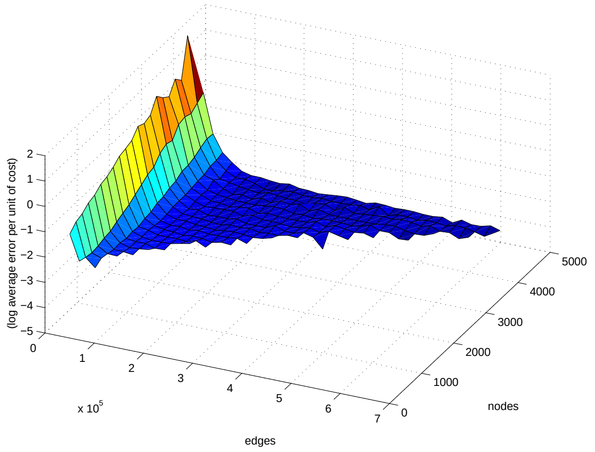


Figure 4: Average error in a shaken grid.

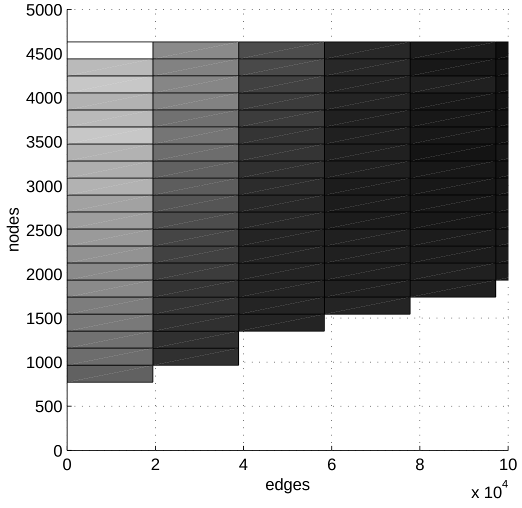


Figure 5: Average error in a shaken grid:top view.

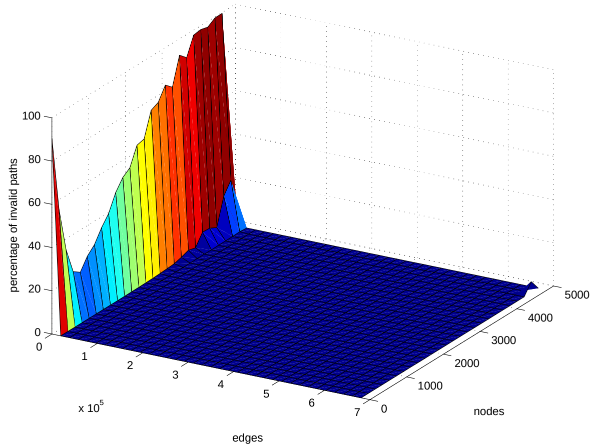


Figure 6: Percentage of failed planning processes in shaken grid.

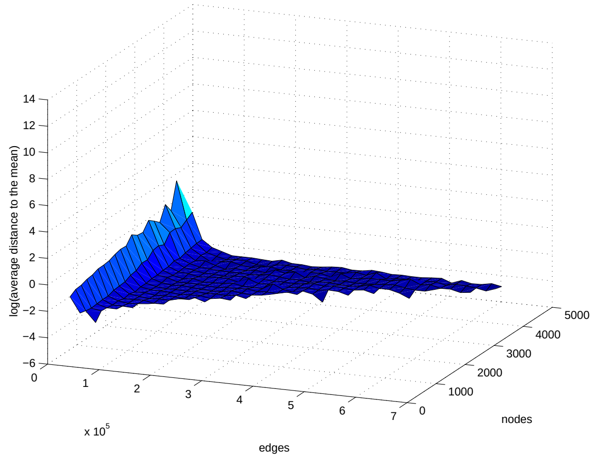


Figure 7: Average distance to the mean in shaken grid.

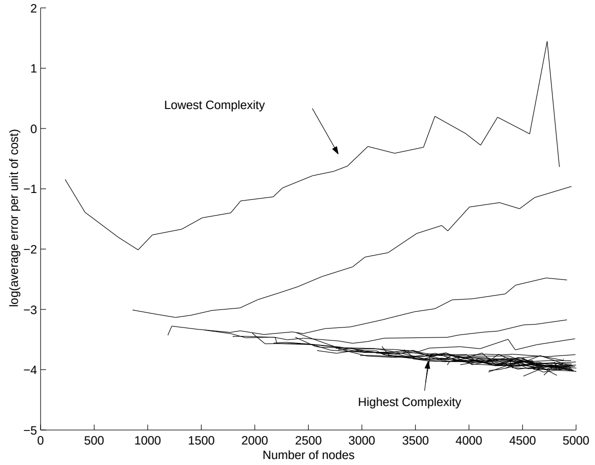


Figure 8: Error for different complexities and varying number of vertices

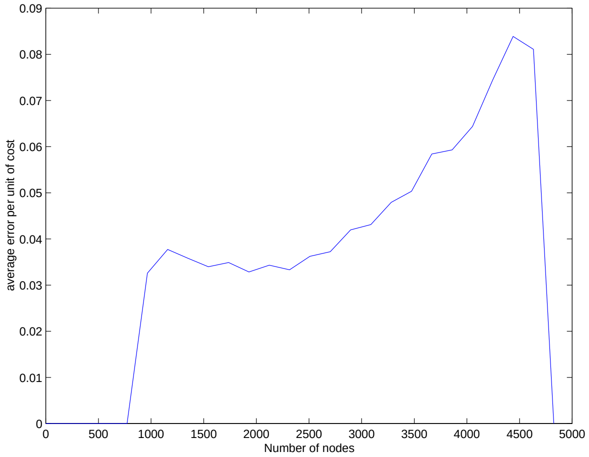
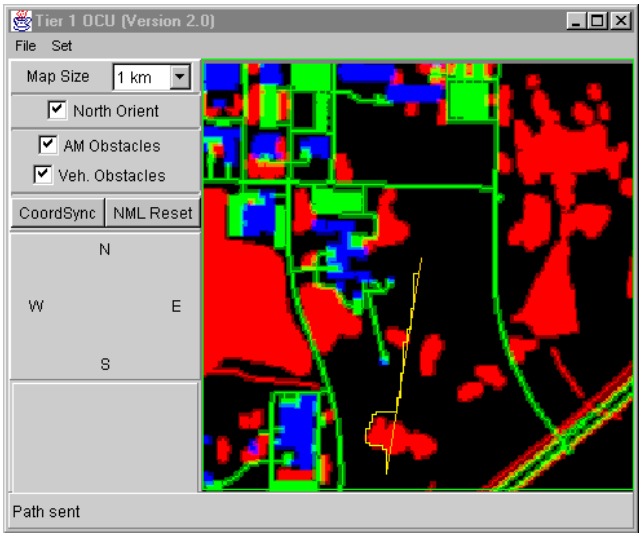


Figure 9: Error for fixed complexity and varying number of vertices



Map Size 1 km

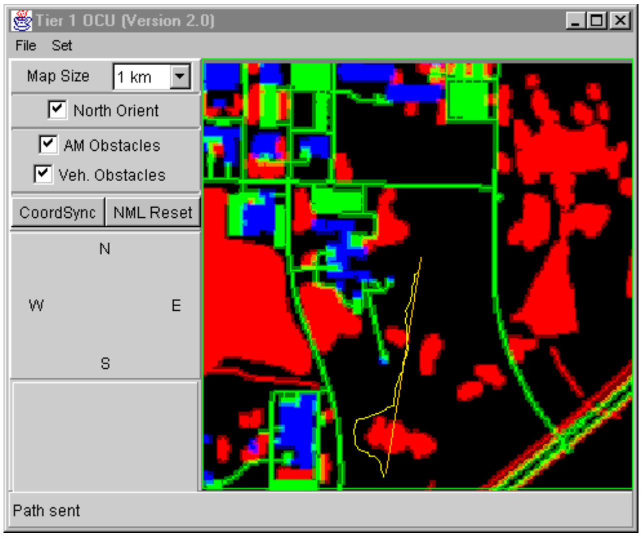
☑ North Orient

☑ AM Obstacles

☑ Veh. Obstacles

Path sent

Figure 10: Planning result for complex cost map with four-connected graph.



Map Size 1 km

☑ North Orient

☑ AM Obstacles

☑ Veh. Obstacles

Path sent

Figure 11: Planning result for complex cost map with many-connected graph.

Vehicle Level Planner [5]. In these experiments, a four-connected graph and a shaken graph of the form of section 2.2 were run using a complex world model and cost function. The four-connected graph had a grid size of 8 meters with 61012 connections and the shaken graph had a grid size of 11 meters with 45086 connections (26% fewer connections) and was shaken ±5.5 meters. The world model contained a priori information on the NIST grounds at 4 meter resolution including the locations of wooded areas, buildings, roads, and fences. It should be noted that the world model resolution is twice that of the four-connected graph and almost three times that of the highly-connected graph.

In the Demo III Vehicle Level Planner, the planning module passes path segment endpoints (the vertices of the planning graph) to the world model for evaluation. The world model simulates driving a straight line path (the edges of the planning graph) between these end points and returns the cost of traversal to the planner. The planner then conducts an optimal search algorithm to find the cheapest path (in reference to the cost function used by the world model). The cost function used by the world model favored paths that avoided roads and buildings, and drove next to, but not in wooded areas combined with the time of traversal of the route (assumed uniform vehicle velocity over the route segment).

The straight line segments used by the world model may cause plan failures when the resolution of the planning graph is less then that of the world model.

This occurs when a very narrow low-cost corridor is surrounded by a very high cost area. It may occur that there are no straight line segments at the graph resolution that traverse this low-cost corridor. This phenomenon can be avoided in the highly-connected graph by adding additional vertices in these high payoff areas. This approach was not taken in the experiments described below.

Using this planning system, we found that the highly-connected graph performed as much as 27% better then the four-connected graph, even though it used 26% fewer connections. Sample output paths may be seen in Figure 10 for the four-connected graph and Figure 11 for the highly-connected graph. A snap-shot of the world model may be seen as the background of these images. As one would expect, the benefit of using the highly-connected graph is directly tied to the shape of the optimal path. For straight paths, the two graphs performed on par with each other. For paths which required many turns, the highly-connected graph significantly outperformed the four-connected graph.

4 Conclusion

- "Optimal" paths found using the four-connected grid based graph are in general, directionally biased, favoring the traversal of the space in certain directions and not in others. They also create symmetries that result in noncommittal paths. Shaken grids and high connectivity between vertices was shown numerically to improve these pitfalls.

- The number of edges in the graph and their cost evaluation are in most cases, the major culprit for computational complexity. Therefore, it is recommended that the graph design process starts by determining the number of edges that can be evaluated, and then selecting the number of vertices that give the least error.

- Numerical evaluation of the error are in most cases the only way to select parameters for the formation of search graphs in complex environments. Most analytical evaluations of the complexity in the literature make the assumption that the burden of computational complexity is in the "opening" of the vertices in the search graph, and are not readily applicable to planning problems.

References

[1] J. Albus. Outline for a theory of intelligence. *IEEE Transactions on Systems, Man, and Cybernetics*, 21:473–509, 1991.

[2] J. Albus, A. Meystel, and A. Lacaze. Multiresolutional planning with minimum complexity. *Intelligent System and Semiotics*, 97.

[3] J.S. Albus. *Brain, Behavior, and Robotics.* McGraw-Hill, 1981.

[4] R.S. Alexander and N.C. Rowe. Path planning by optimal-path-map construction for homogenous-cost two-dimensional regions. In *IEEE International Conference on Robotics and Automation - 1990*, 1990.

[5] Stephen Balakirsky and Alberto Lacaze. World modeling and behavior generation for autonomous ground vehicles. In *Proceedings IEEE International Conference on Robotics and Automation*, 2000.

[6] J. Bondy and U. Murty. *Graph Theory with Applications.* North Holland, 1976.

[7] N. Deo. *Graph Theory with Applications to Engineering and Computer Science.* Prentice Hall, 1974.

[8] M. Garey and D. Johnson. *Computers and Intractability.* Freeman, 1979.

[9] F. Harary. *Graph Theory.* Addison Wesley, 1972.

[10] Y. Maximov and A. Meystel. Optimum design of multiresolutional hierarchical control systems. In *Proceedings of IEEE Int'l Symposium on Intelligent Control*, pages 514–520, Glasgow, U.K., 1992.

[11] C.H. Papadimitriou. *Computational Complexity.* Addison Wesley, 1994.

[12] F.P. Preparata and M.I. Shamos. *Computational Geometry, An Introduction.* Springer Verlag, 1988.

[13] J.H. Reif. Complexity of the generalized moving problem. In et al Schwartz, editor, *Planning Geometry and Complexity of Robot Motion.* Ablex Publishing Corporation, 1987.

[14] J.T. Schwartz, M. Sharir, and J. Hopcroft, editors. *Complexity of the Generalized Moving Problem.* Ablex Publishing Corporation, 1987.

Metrics for Embedded Collaborative Intelligent Systems[1]

Michael E. Cleary, Mark Abramson, Milton B. Adams, Stephan Kolitz
Draper Laboratory
555 Technology Square, MS 3F, Cambridge, MA 02139
{mcleary, mabramson, adamsm, kolitz}@draper.com

ABSTRACT

The intelligence of a network of agents is reflected in the complexity of missions that can be accomplished, the degree of coordination/cooperation among the agents, and the level of uncertainty the system can tolerate and still accomplish its missions. The networked system must be able to evaluate a situation, devise an appropriate response, and act accordingly. Metrics must be devised to capture the complexity and surprises of the real world, and to capture the system's need to reason about its situation so as to uncover unanticipated problems and opportunities. Inputs for developing autonomous capability specifications (and thus metrics of interest) include (1) descriptions of expected missions, (2) the space of mission parameters, and (3) the cost/benefitratio for operational concepts. These inputs come from both current and anticipated missions. Several of our recent projects have sought to quantify operational metrics for autonomous ground, air and undersea vehicles. This paper presents our approach to high-level design of autonomous vehicles that produces the three inputs for metric development. The approach and parameter spaces are illustrated with examples derived from several vehicle projects.

Keywords: metrics, intelligence quotient, intelligent systems, autonomous systems, collaborative systems, situation awareness, planning under uncertainty, orders of intelligence.

1 INTRODUCTION

The intelligence of a network of agents is a complex characteristic that can be quantified and measured in a wide variety of ways. Our work on the design of intelligent autonomous vehicles and programs to develop such vehicles has made clear that the type of metric we develop will be chosen to meet a particular objective. For instance, commercial sponsors will likely optimize some functionality, while researchers may try to optimize some measure of "pure" intelligence. After reviewing a number of systems in ground, air and undersea domains, it becomes clear that the major characteristics of intelligence for any complex set of vehicles are the broad areas of multi-vehicle collaboration, understanding the world they operate in (situation awareness) and responding appropriately (planning under uncertainty).

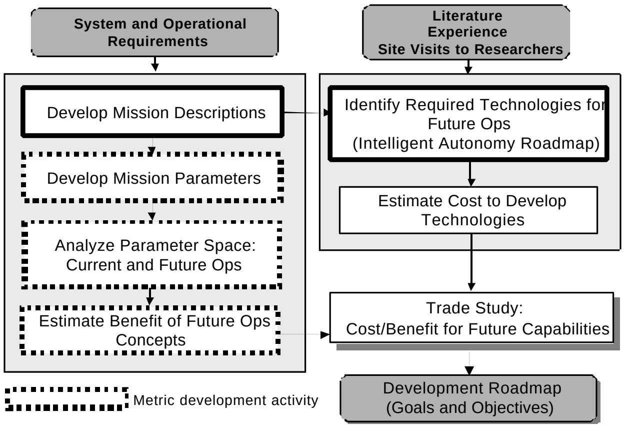

System and Operational Requirements
Literature
Experience
Site Visits to Researchers
Develop Mission Descriptions
Develop Mission Parameters
Analyze Parameter Space: Current and Future Ops
Estimate Benefit of Future Ops Concepts
Identify Required Technologies for Future Ops (Intelligent Autonomy Roadmap)
Estimate Cost to Develop Technologies
Trade Study: Cost/Benefit for Future Capabilities
Development Roadmap (Goals and Objectives)
Metric development activity

Figure 1 -- Development Roadmap

To guide the design of intelligent vehicles for particular domains, we have used the process illustrated in Figure 1. There are two major efforts shown – the left column focuses on the missions the vehicle is intended to accomplish, while the right column focuses on the technologies required to accomplish those missions. The two columns could be loosely labeled requirements pull and technology push, respectively. The areas we have considered for metric analysis to date are those shown surrounded with dotted lines. Once thorough descriptions of the vehicles' missions are developed, those are reviewed to extract parameters that affect performance. The mission descriptions are then extended to probe the space of the identified parameters. This process is illustrated in detail in Section 2.

A more humanly intuitive representation of the parameter space was sought, since the bare listing of parameters can be daunting (Section 3). This introduces significant subjectivity, but allows aspects of intelligence to be clustered that seem to lead to strong collaborative systems.

Section 4 discusses an attempt to quantize intelligence into "orders" of intelligence. It begins with the point that "intelligence" is still a relatively undefined area, needing substantial work in the component technologies and in the development of appropriate metrics. Despite that reservation, candidate levels of intelligence capability are described that might serve as an IQ for autonomous systems.

Costs are another aspect of intelligence that require attention and metrics (Section 5). For instance, a sponsor may seek to develop a comprehensive technology roadmap that will determine what technologies need investment to meet a particular set of system and operational requirements.

The paper concludes with a brief discussion of some future directions for our work (section 6) and a summary (section 7).

2 CONSTRUCTION OF PARAMETER SPACE

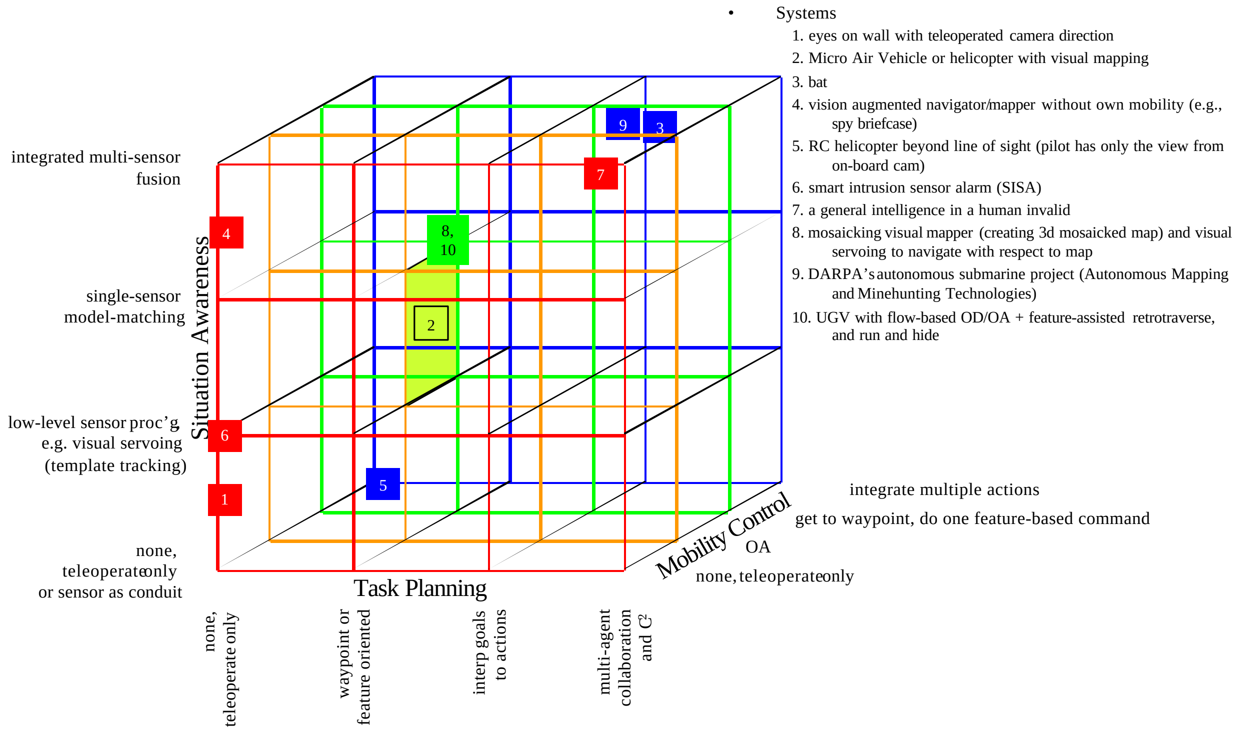
The simplest way to evaluate a system's success or failure at its task is often binary – did it accomplish some goal? For instance, in RoboCup Soccer [3] as in human games, a single score is the final arbiter of success. However, the single score does not capture the complexity of the domain or of the team's approach to various elements of the problem. Thus additional "scores" are developed that rate game players on the skills that contribute to the final game score. Such more detailed scores can be combined into a single weighted score, using multi-objective optimization techniques [1,2]. However, that requires significant work to determine appropriate weightings and combination techniques.

The first step toward such a development is to flesh out the parameter space of the task. A large number of factors can be considered in a thorough analysis of a collaborative group of vehicles. We use the three characteristic areas named above (collaboration, situation awareness, and planning under uncertainty). The following incomplete lists indicate some of the important elements for robots facing dangerous situations (military or other). Each metric on the list requires a range of acceptable values and a weighting factor for combining them with other components. The factors can then be processed to produce a combined metric if such a score is desired.

- **Multi-vehicle collaboration factors**
  - number of interacting agents
  - degree of coordination/cooperation among the agents
  - degree of improvement in situation awareness due to multiple vehicles
  - success of dynamic replanning to maintain configuration for communication
- **Situation awareness**
  - amount of complexity and surprise of real world captured
  - number of elements
  - level of interactions between elements
  - dynamism
  - model complexity for target identification
  - observability
  - environmental challenge
    - + clear air/daylight – to – storms at night
    - + desert (all is visible) – to – mountainous (hard to see details)
    - + textured (landmarks differ) – to – desert/no texture
  - threat types
    - + from known type/location – to – suspected – to – unknown till aggression
    - + from id is straightforward (e.g., surface-to-air-missile (SAM) radar) – to – difficult/uncertain (visual or synthetic aperture radar (SAR), near friendlies, signature similar to neutral or friendly
  - neutrals
    - + known type/location – to – threats masquerading as neutrals
  - friendlies
    - + known type/location – to – identify-friend-foe (IFF) transponders off/broken or known but near threats
  - navigation
    - + sensors functioning and low uncertainty – to – sensors dropping out/damaged or high uncertainty
  - vehicle state (including equipage)
    - + sophistication of health monitoring and reconfiguration
  - time to sense and assimilate (separate from time to plan)
    - + enough time – to – insufficient time due to tempo or number of targets (so need to prioritize sensing and assimilation)
  - can successfully identify a target
  - can detect environmental changes of the following types:
    - + threats
    - + terrain
    - + collision
    - + targets of opportunity
- **Decision making and executing under uncertainty**
  - extent that system reasons about its situation
    - + uncovers unanticipated problems
    - + uncovers opportunities
  - level of uncertainty the system can tolerate
  - performs under available time to plan

- dynamic time constant that system can reason within
- stochasticity - number of contingencies handled by system
- number of decisions (i.e., size of planning problem)
- quality of plan generation / selection algorithms
- quality of planning approach (algorithms and representations)
- complexity of mission / problem
- complexity of controllable system
- number of plan elements in flux simultaneously
- number of levels in planning problem
- ability to perform dynamic replanning due to:
  - + change in mission objectives
  - + environmental change detected

Such a list of parameters is daunting, and only becomes more difficult to grasp and synthesize as the level of detail grows. A more intuitive representation was sought to support analysis of the trade-offs involved in system design and funding. The result is discussed in the following section.



integrated multi-sensor fusion
single-sensor model-matching
low-level sensor proc'g e.g. visual servoing (template tracking)
none, teleoperate only or sensor as conduit
Situation Awareness
Task Planning
none, teleoperate only
waypoint or feature oriented
interp goals to actions
multi-agent collaboration and C²
Mobility Control
integrate multiple actions
get to waypoint, do one feature-based command
OA
none, teleoperate only
1
2
3
4
5
6
7
8, 10
9

- Systems
  1. eyes on wall with teleoperated camera direction
  2. Micro Air Vehicle or helicopter with visual mapping
  3. bat
  4. vision augmented navigator/mapper without own mobility (e.g., spy briefcase)
  5. RC helicopter beyond line of sight (pilot has only the view from on-board cam)
  6. smart intrusion sensor alarm (SISA)
  7. a general intelligence in a human invalid
  8. mosaicking visual mapper (creating 3d mosaicked map) and visual servoing to navigate with respect to map
  9. DARPA's autonomous submarine project (Autonomous Mapping and Minehunting Technologies)
  10. UGV with flow-based OD/OA + feature-assisted retrotraverse, and run and hide

Figure 2 -- Three-Dimensional Intelligence Space

3 Graphical Parameter Space

A three-dimensional graphical approach was used to illustrate where various systems and system designs fell in the overall parameter space (Figure 2). This shows a particular three axes in the parameter space, recognizing that the whole estimation and metrics space is highly multi-dimensional. Several such charts were prepared, but no canonical axes were identified that best serve all analysis purposes for all autonomous systems. The figure shows axes of situation awareness, mobility, and task planning as creating a 3D intelligence space. A variety of autonomous and non-autonomous systems are included in the figure to highlight key parts of the resulting space.

The representation's key weakness is inherent in the choice of any set of 3 dimensions– key information from a fuller, higher-dimensional space is lost. Also problematic is the apparent linearity between ticks along any axis – what conclusions can be drawn by systems shown N ticks apart? Still, there is the strong sense that this captures something fundamental and accurate about the intelligence present in a variety of compared systems. The primary difficulty with this approach, however, remains the subjective judgement that only a small number of axes is enough to grasp the entire intelligence space.

4 AN INTELLIGENCE QUOTIENT?

We have been asked at various junctures to provide metrics for autonomous systems development, in a similar vein to those provided by (for instance) engineers working in other disciplines who do not hesitate to propose metrics. That has been a difficult request to answer, until the various exercises reported above led us to a key conclusion: Mature technologies can support more precise performance targets than immature technologies. For instance, a group researching automatic target recognition (ATR) can aim to decrease the false alarm rate by 5%. However, what similar metric applies to the broader aim of "increase intelligent autonomy"?

This section discusses a reservation about characterizing intelligence, then proposes levels of capability that are our best-yet "intelligence quotient" for autonomous systems.

4.1 A Philosophical Reservation

Answering the above question may depend on how the question is phrased, but consider this goal: enable autonomous dynamic mission replanning, based on discovered targets and conditions expected in the target area, while out of communication with the human operator. Several questions spring to mind. What technologies apply? What are their margins for improvement? Do we even know what is necessary to achieve the goal? One approach is to consider finer-grained technologies rather than the broad term of "autonomy". For instance, the following appear more susceptible to metrification.

- Decrease route planning time-to-plan by 20% given contingencies of type A.
- Increase ATR reliability for particular target/environment pairs by 10%.
- Increase situation recognition capability by increasing contingency representation flexibility by 10 times.

We conclude that "intelligent autonomy" is an immature "technology" that is actually a composition of underlying technologies, all of varying maturities. A small set of examples of component technologies with clear deficiencies (compared to human-level capabilities) follows.

- Sensor data interpretation
- Situation awareness and assessment
- Communication
  - Efficient – perhaps better named "data communication" (bandwidth, rates, etc)
  - Effective – perhaps better named "knowledge communication" (content, concepts, transparency of thought processes)
- Knowledge representation – know, represent and share:
  - What data toward what goals in what timeframes?
  - Why does datum A or set of data B matter?
  - Timeliness of concern
    - + Damage is expected to occur by time T (e.g., hostile strike group detected headed for barrier)
    - + Unless used by time T, data C not useful (e.g., a moving surface-to-air-missile launcher is detected 1 mile from bunker moving 10 mph - must use information within 6 minutes)
  - Relatedness of data
- Collaboration
    - + Understand others' goals
    - + Infer intent from observed behavior

Thus finding ways to divide intelligence and autonomy into appropriate sub-technologies that can be weighed and combined properly is a critical problem facing this effort. Lacking such a reliable analysis tool, we next consider one way to approach its formulation.

4.2 Orders of Intelligence

Given the above reservation, let us proceed to characterize intelligence by asking: how hard is a planning and execution problem? Time to plan (TTP) depends on the size of the planning problem, but Moore's Law will reduce TTP significantly by increasing the feasible size of planning problems. However, TTP also depends on (a) the planning approach (algorithms and representations) and (b) the problem complexity. Size of the problem is the easiest to provide metrics for. The other two factors are used to modulate the metrics. If a planning agent is only concerned with a certain time horizon (e.g., 10 milliseconds, 1 hour, 1 day), the level of detail it considers is similarly bounded. Thus planning problems can be of similar sizes whether at the level of a single vehicle or a fleet of vehicles.

There are numerous planning approaches. For well-characterized and well-formulated domains, search in a pre-defined state space is satisfactory. For other problems, current pure research effots are unable to provide a well-defined solution. More pragmatically, planning and execution systems can use a variety of hybrid approaches, the integration of which pose at least engineering issues.

Problem complexity addresses characteristics beyond the simple size of the problem. The characteristics that make planning, estimation and control difficult include the following elements. Since planning needs to be concerned with what can be expected to occur, it must be concerned with expected results from estimation and control, that are affected by the following elements.

- observability – the degree of hidden state (in controlled system or in situation being monitored)
- complexity of the controllable system. E.g., number and type of actuators, static and dynamic stability of the vehicle.
- situation awareness complexity. E.g.:
  - number of elements
  - interactions between elements
  - dynamism (e.g., likelihood to loose lock in tracking subsystem)
  - model complexity for target identification (e.g., 2D image templates, 3D shape, functional analysis based on shape, behavioral)
  - degree to which situation awareness (SA) fulfills expectations
- number of interacting agents. Especially if multiple agents are simultaneously planning
- number of plan elements in flux simultaneously. E.g., (a) is plan in place before SA is received, or (b) is SA being integrated while plan using it is being created? Regarding example (a) consider the plan “go to area X and find tanks” (where “tanks” will be bound to those found by SA), whereas for (b) consider what the system needs to do when it finds itself unexpectedly under attack from unknown quarters.
- number of levels in planning problem due to (i) number of elements, (ii) number of time horizons, etc.

One approach to creating metrics for these problems is to classify problems from the domain into nominal orders of difficulty, then set targets for various demonstrations which move along the spectrum of difficulty. For instance, reasonable goals might be created by aiming to solve a problem in 1 second in each demo year, where the size and complexity of the problem increases over time. Based on the nominal characterization below of levels of difficulty, the solvable problem size could increase from 10^0 in demo 1 (say year 2), to 10^1 in demo 2 (year 4), and 10^2 in demo 3 (year 6). This folds together the expected advances in processor speed and capacity embodied in Moore’s Law with improvements in planning approaches resulting from pure and applied research progress. Table 1 captures this approach and leaves space for additional metrics at various levels of maturity.

	10^0	10^1	10^2	10^3
Demo 1	1 second (TTP0)			
Demo 2		1 second (TTP1)		
Demo 3			1 second (TTP2)	
Beyond				1 second (TTP3)

Table 1 -- Problem Size, and Plan for Increasing Demonstrable Complexity

4.3 Nominal candidate orders of intelligence

The following lists indicate relative order of magnitude capabilities that could be grouped together to assess the maturity of a system’s intelligence. These are illustrative, not final. Order 0 activities may exist in preliminary commercial research forms or may need applied research and engineering to be fielded. Higher order activities are believed to be beyond the current state of the art.

Order 0 activities:

- Single vehicle plans including (a) multi-waypoint path planning and execution cognizant of known threats, (b) obstacle avoidance given some warning, (c) deck landing in relatively benign environment
- Multiple vehicle plans, for non-interacting vehicles
- Plan to search area of regard (AOR) for target, where AOR is essentially flat and open, and target can be found by template matching.
- Re-plan communication relay service due to disruption of channel, using prior known assets.
- Re-plan for changed objective, where accomplishment of the objective is in the future from the current time-horizon.
- Re-plan task particulars due to change in SA. E.g., arrive in kill box and discover that the targets to be hit are tanks instead of a column of trucks.
- identify targets of opportunity based on their appearance

Order 1 activities:

- single vehicle obstacle avoidance given less warning and/or more constraints on response (e.g., in confined airspace due to terrain or other vehicles, near vehicle limits for responsiveness)

- single vehicle deck landing in moderate sea state and/or moderate visibility
- Plan to act as autonomous communication relay between moving communication partners, where the partners are moving in ways that are expected to disrupt communication within foreseeable future. Thus plan must include a plan to identify and involve additional communication relays. Alternative contingencies would include planning for disruptions that might occur due to weather, jamming, or other hostile activity.
- Re-plan for changed objective, where accomplishment is within current time-horizon, requiring current SA to be integrated while planning is underway using the being-acquired SA.
- identify targets of opportunity based on their appearance where (e.g.) detection depends on sensor angle, so vehicle must do more extensive search to cover the space of AOR-cross-sensor-attitude. E.g., tanks at edge of forest need to be sensed from the open side. Vehicle should understand the constraints (not just fly more lanes of a survey pattern).
- multi-vehicle plans for interacting vehicles
- strike group flight plan through waypoints and around known threats
- re-plan task goals due to change in SA. E.g., while on wild weasel mission switch to coordinated multi-vehicle SAM attack.

Order 2 activities:

- single vehicle deck landing in high sea state and/or low visibility and/or high and gusty winds
- coordinated obstacle avoidance for a strike group flying very close together

Order 3 activities:

- identify targets of opportunity based on their behavior (from prior planning/SA need model of behavior and identification of behavior based on model)

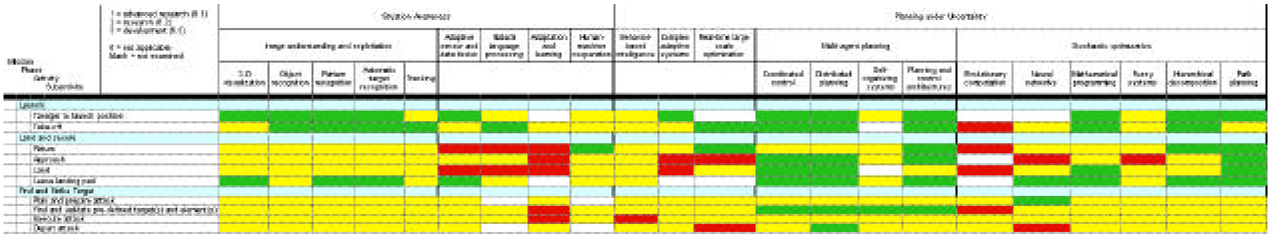
These characterizations build on those detailed in Section 2 – as vehicles increase their ranking in the Orders of Intelligence, they exhibit more capability in the parameter spaces. For example, consider the multi-vehicle collaboration factor of **degree of coordination / cooperation among the agents**. The Order 0 system includes *multiple vehicle plans for non interacting vehicles*. This could include an system that distributes the team goals among the individual agents for separate completion. The Order 1 system includes a higher level capability in this area of *multi-vehicle plans for interacting vehicles*. Here agents can communicate to one another when they fail or if they are able to take on an increased set of tasks. The Order 2 system increases the requirements on coordination and cooperation to *coordinated obstacle avoidance for a strike group flying very close together*. The system will be required to share situation awareness information and plan coordinated responses at very short time constants.

5 METRICS FOR COSTS

To create a plan for funding toward a goal, an assessment must be made of the state of the technology against the require capabilities. Figure 3 shows such an assessment. It was constructed by asking technology experts to determine the state of maturity of their technologies for solving various parts of a vehicle's parameter space. The colors indicate technological maturity levels:

red	pure research needed (6.1)
yellow	applied research needed (6.2)
green	ready for engineering (6.3)
blank	not applicable

Although this is not a measure of intelligence per se, it supports analyses leading to the construction of intelligence vehicles and groups of vehicles.



Mission Phase Activity Subactivity	Situation Awareness									Planning under Uncertainty												
| = advanced research (6.1) | = research (6.2) | = development (6.3) X = not applicable blank = not examined	Image understanding and exploitation					Adaptive sensor and data fusion	Natural language processing	Adaptation and learning	Human-machine cooperation	Behavior-based intelligence	Complex adaptive systems	Real-time large scale optimization	Multi-agent planning				Stochastic optimization					
	3-D visualization	Object recognition	Pattern recognition	Automatic target recognition	Tracking								Coordinated control	Distributed planning	Self-organizing systems	Planning and control architectures	Evolutionary computation	Neural networks	Mathematical programming	Fuzzy systems	Hierarchical decomposition	Path planning
Launch																						
Navigate to launch position	🟩	🟩	🟩	🟩	🟨	🟩	🟨		🟨	🟨	🟩		🟩	🟩	🟨	🟩			🟩	🟨	🟩	🟩
Takeoff	🟨	🟩	🟩	🟩	🟩	🟨	🟩	🟨	🟨	🟨	🟨	🟩	🟩	🟩		🟩	🟥	🟨	🟩	🟨	🟩	🟨
Land and recover																						
Return	🟨	🟨	🟨	🟨	🟨	🟥	🟥	🟥	🟩	🟨	🟨	🟩	🟨	🟨	🟨	🟩	🟥	🟨	🟨	🟨	🟩	🟩
Approach	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟥	🟨	🟨	🟥	🟥	🟩	🟩	🟨	🟩		🟥	🟨	🟥	🟨	🟩
Land	🟨	🟨	🟨	🟨	🟨	🟥	🟥	🟥	🟨	🟨	🟥	🟨	🟩	🟩		🟨	🟥	🟨	🟩	🟨	🟨	🟩
Leave landing pad	🟩	🟨	🟩	🟩	🟨	🟩	🟨		🟨	🟨			🟩	🟩	🟨	🟩		🟩	🟩	🟨	🟩	🟩
Find and Strike Target																						
Plan and prepare attack	🟨	🟨	🟨	🟨	🟨	🟨		🟨		🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟩	🟨	🟨	🟨	🟨
Find and validate pre-defined target(s) and element(s)	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟥	🟨	🟨	🟨	🟨	🟩	🟩	🟩	🟩	🟥	🟨	🟨	🟨	🟨	🟨
Execute attack	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟥	🟨	🟥	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨
Depart attack	🟨	🟨	🟨	🟨	🟨	🟨		🟨		🟨	🟨	🟥	🟨	🟩	🟨	🟨	🟨	🟥	🟨	🟨	🟨	🟨

Figure 3 -- Technology Roadmap (partial).
Columns are technologies considered appropriate for addressing the domain, while rows are elements of the vehicle's parameter space.

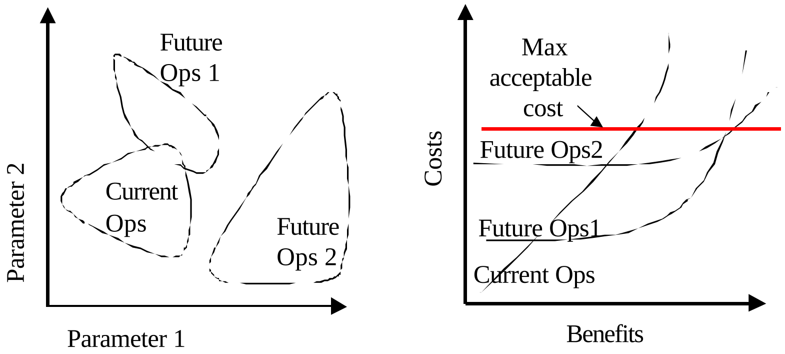


Figure 4 -- Cost/Benefit Analysis.
(left) Where in the parameter space future intelligently autonomous operations exist.
(right) Part of a cost-benefit analysis to select among various operations based on cost.

The notional charts in Figure 4 illustrate how required capabilities can be mapped against mission descriptions of current and future operations, to help determine which are more valued, and to help determine which are expected to be more expensive. Formal methods for such cost projections would be very helpful.

6 Future Directions

Substantial work has been done in applying valuations to multi-attribute (multi-criteria) problems. Besides a number of good textbooks (e.g., [1,2]), various techniques have been formalized to assist in this process. We intend to extend the work reported here by investigating and applying formal tools to the domain characteristics discussed above.

7 Summary

The intelligence of an autonomous vehicle is a complex multi-dimensional characteristic evaluated in a wide variety of dynamic situations, for which no obvious algorithmic measures exist. Several attempts to analyze system complexity and intelligence have been presented in this paper that are drawn from work done for recent and current projects working toward intelligent autonomous vehicles. These analyses have sought to uncover the collaboration, planning and situational awareness challenges facing an autonomous vehicle in difficult conditions, to assist engineers and sponsors in focusing project efforts. Although the analyses reported here have been useful first steps toward the significantly complex vehicles imagined, more work is clearly required before intelligence and intelligent systems can be automatically analyzed and measured.

References

1. Clemen, Robert T., *Making Hard Decisions: An introduction to decision analysis*, PWS-Kent Publishing Co., Boston, 1991

2. French, Simon, *Decision Theory: An introduction to the Mathematics of Rationality*, Ellis Harwood, Ltd. Chichester, West Sussez, England, 1986

3. RoboCup-97: Robot Soccer World Cup I, Springer-Verlag, 1998.

Evolution of Mobile Agents

Timothy K. Shih
Multimedia Information NEtwork (MINE) Lab
Department of Computer Science and Information Engineering
Tamkang University
Tamsui, Taipei Hsien, Taiwan 251, R.O.C.
email: TSHIH@CS.TKU.EDU.TW

Abstract

Mobile agents are powerful. A mobile agent can travel on the Internet, perform tasks, and report to its owner the achievement. Mobile agent techniques are used in E-commerce, distributed applications, distance learning, and others. However, it is hard to find a strategic method, which tells how mobile agents should behave on the Internet. In this paper, we propose such a mechanism. Based one the concepts of Food Web, one of the laws that we may learn from the natural besides neural networks and genetic algorithms, we propose a theoretical computation model for mobile agent evolution on the Internet. We define an agent niche overlap graph and agent evolution states. We also propose a set of algorithms, which is used in our multimedia search programs, to simulate agent evolution. Agents are cloned to live on a remote host station based on three different strategies: the brute force strategy, the semi-brute force strategy, and the selective strategy. Evaluations of different strategies are discussed. Guidelines of writing mobile agent programs are proposed. The technique can be used in distributed information retrieval which allows the computation load to be added to servers, but significantly reduces the traffic of network communication.

1 Introduction

Mobile agents are software programs that can travel over the Internet. Mobile search agents find the information specified by its original query user on a specific station, and send back search results to the user. Only queries and results are transmitted over the Internet. Thus, unnecessary transmission is avoided. In other words, mobile agent computing distributes computation loads among networked stations and reduces network traffic.

The environment where mobile agents live is the Internet. Agents are distributed automatically or semi-automatically via some communication paths. Therefore, agents meet each other on the Internet. Agents have the same goal can share information and cooperate. However, if the system resource (e.g., network bandwidth or disk storage of a station) is insufficient, agents compete with each other. These phenomena are similar to those in the ecosystem of the real world. A creature is born with a goal to live and reproduce. To defense their natural enemies, creatures of the same species cooperate. However, in a perturbation in ecosystems, creatures compete with or even kill each other. The natural world has built a law of balance. Food web (or food chain) embeds the law of creature evolution. With the growing popularity of Internet where mobile agents live, it is our goal to learn from the natural to propose an agent evolution computing model over the Internet. The model, even it is applied only in the mobile agent evolution discussed in this paper, can be generalized to solve other computer science problems. For instance, the search problems in distributed Artificial Intelligence, network traffic control, or any computation that involves a large amount of concurrent/distributed computation. In general, an application of our Food Web evolution model should have the following properties:

- The application must contain a number of concurrent events.
- Events can be simulated by some processes, which can be partitioned into a number of groups according to the properties of events.
- There must exists some consumer-producer relationships among groups so that dependencies can be determined.
- The number of processes must be large enough.

For instance, with the growing popularity of Internet, Web-based documentation are retrieved via some search engine. Search processes can be conducted as several concurrent events distributed among Internet stations. These search events of the same kind (e.g., pursuing the same document) can be formed in a group. Within these agent groups, search agents can provide information to each other. Considering the amount of Web sites in the future, the quantity of concurrent search events is reasonably large.

We have surveyed articles in the area of mobile agents, personal agents, and intelligent agents. The related works are discussed in section 2. Some terminologies and definitions are given in section 3, where we also introduce the detail concepts of agent communication network. In our model, an agent evolves based on state transitions, which are also discussed. A graph theoretical model describes agent dependencies and competitions is also given. Agent evolution computing algorithms are addressed in section 4. And finally, we discuss our conclusions in section 5.

2 Related Works

The concept of mobile agent is discussed in several articles [3, 4]. Agent Tcl, a mobile-agent system providing navigation and communication services, security mechanisms, and debugging and tracking tools, is proposed in [1]. The system allows agent programs move transparently between computers. A software technology called Telescript, with safety and security features, is discussed in [7]. The mobile agent architecture, MAGNA, and its platform are presented in [3]. Another agent infrastructure is implemented to support mobile agents [4]. A mobile agent technique to achieve load balancing in telecommunications networks is proposed in [6]. The mobile agent programs discussed can travel among network nodes to suggest routes for better communications. Mobile service agent techniques and the corresponding architectural principles as well as requirements of a distributed agent environment are discussed in [2].

3 Definitions

Agents communicate with each other since they can help each other. For instance, agents share the same search query should be able to pass query results to each other so that redundant computation can be avoided. An *Agent Communication Network* (ACN) serves this purpose. Each node in an ACN represents an agent on a computer network node, and each link represents a logical computer network connection (or an agent communication link). Since agents of the same goal want to pass results to each other, agent communication relations can be described in a complete graph. Therefore, an ACN of agents hold different goals is a graph of complete graphs. Since agents can have multiple goals (e.g., searching based on multiple criteria), an agent may belong to different complete graphs.

We define some terminologies used in this paper. A *host station* (or *station*) is a networked workstation on which agents live. A *query station* is a station where a user releases a query for achieving a set of goals. A station can hold multiple agents. Similarly, an agent can pursue multiple goals. An *agent society* (or *society*) is a set of agents fully connected by a complete graph, with a common goal associated with each agent in the society. A goal belongs to different agents may have different priorities. An agent society with a common goal of the same priority is called a *species*. Since an agent may have multiple goals, it is possible that two or more societies (or species) have intersections. A *communication cut set* is a set of agents belong to two distinct agent societies, which share common agents. The removing of all elements of a communication cut set results in the separation of the two distinct societies. An agent in a communication cut set is called an *articulation agent*. Since agent societies (or species) are represented by complete graphs and these graphs have communication cut sets as intersections, articulation agents can be used to suggest a shortest network path between a query station and the station where an agent finds its goal. Another point is that an articulation agent can hold a *repository*, which contains the network communication statuses of links of an agent society. Therefore, network resource can be evaluated when an agent checks its surviving environment to decide its evolution policy.

An agent evolves. It can react to an environment, respond to another agent, and communicate with other agents. The evolution process of an agent involves some internal states. An agent is in one of the following states after it is born and before it is killed or dies of natural:

- **Searching**: the agent is searching for a goal
- **Suspending**: the agent is waiting for enough resource in its environment in order to search for its goal
- **Dangling**: the agent loses its goal of surviving, it is waiting for a new goal
- **Mutating**: the agent is changed to a new species with a new goal and a possible new host station

An agent is born to a *searching state* to search for its goal (i.e., information of some kind). All creatures must have goals (e.g., search for food). However, if its surviving environment (i.e., a host station) contains no enough resource, the agent may transfer to a *suspending state* (i.e., hibernation of a creature). The searching process will be resumed when the environment has better resources. But, if the environment is lack of resources badly (i.e., natural disasters occur), the agent might be killed. When an agent finds its goal, the agent will pass the search results to other agents of the same kind (or same society). Other agents will abort their search (since the goal is achieved) and transfer to a *dangling state*. An agent in a dangling state can not survive for a long time. It will die after some days (i.e., a duration of time). Or, it will be re-assigned to a new goal with a possible new host station, which is a new destination where the agent should travel. In this case, the agent is in a *mutating state* and is reborn to search for the new goal. Agent evolution states keep the status of an agent. In order to maintain the activity of agents, in a distributed computing environment, we use message passing as a mechanism to control agent state transitions.

Agents can suspend/resume or even kill each other. We need a general policy to decide which agent is killed. By our definition, a species is a set of agents of the same goal with a same priority. It is the priority of a goal we base on to discriminate two or more species.

We need to construct a direct graph which represents the dependency between species. We call this digraph an *species food web* (or *food web*). Each node in the graph represents a species. All species of a connected food web (i.e., a graph component of the food web) are of the same goal with possibly different priorities. We assume that, different users at different host stations may issue the same query with different priority. Each directed edge in the food web has an origin represents a species of a higher goal priority and has a terminus with a lower priority. Since an agent (and thus a species) can have multiple goals which could be similar to other agents, each goal of an articulation agent should have an associated food web. Therefore, the food web is used as a competition base of agents of the same goal in the same station.

Each food web describes goal priority dependencies of species. Form a food web, we can further derive an *niche overlap graph*. In an ecosystem, two or more species have an *ecological niche overlap* (or *niche overlap*) if and only if they are competing for the same resource. A *niche overlap graph* can be used to represent the competition among species. The niche overlap graph is used in our algorithm to decide agent evolution policy and to estimate the effect when certain factors are changed in an agent communication network. Based on the niche overlap graph, the algorithm is able to suggest strategies to re-arrange policies so that agents can achieve their highest performance efficiency. This concept is similar to the natural process that recover from perturbations in ecosystems.

4 Agent Evolution Computing

The algorithms proposed in this section use the agent evolution states and the niche overlap graphs discussed for agent evolution computing. An agent wants to search for its goal. At the same time, since the searching process is distributed, an agent wants to find a destination station to clone itself. Searching and cloning are essentially exist as a *co-routing relation*. A co-routine can be a pair of processes. While one process serves as a producer, another serves as a consumer. When the consumer uses out of the resource, the consumer is suspended. After that, the producer is activated and produces the resource until it reaches an upper limit. The producer is suspended and the consumer is resumed. In the computation model, the searching process can be a consumer, which need new destinations to proceed search. On the other hand, the cloning process is a producer who provides new URLs.

Agent evolution on the agent communication network is an asynchronous computation. Agents live on different (or the same) stations communicate and work with each other via agent messages. The searching and the cloning processes of an agent may run as a coroutine on a station. However, different agents are run on the same or separated stations concurrently. We use a formal specification approach to describe the logic of our evolution computation. Formal specifications use first order logic, which is precise. In this paper, we use the Z specification language to describe the model and algorithms.

Each algorithm or global variable in our discussion has two parts. The expressions above a horizontal line are the signatures of predicates, functions, or the data types of variables. Predicates and functions are constructed using quantifiers, logic operators, and other predicates (or functions). The signature of a predicate also indicate the type of its formal parameters. For instance, $Agent \times Goal \times Host\_Station$ are the types of formal parameters of predicate $Agent\_Search$. The body, as the second part of the predicate, is specified below the horizontal line.

We use some global variables through the formal specification. The variable $goal\_achieved$ is set to $TRUE$ when the search goal is achieved, $FALSE$ otherwise. We also use two watermark variables, α and β, where α is the basic system resource requirement and β is the minimal requirement. Note that, α must be greater than β so that different levels of treatment are used when the resource is not sufficient.

Global Variables and Constants

$goal\_achieved : Goal\_Achieved$
$\alpha : REAL$
$\beta : REAL$

$\alpha > \beta$

Algorithm $Agent\_Search$ is the starting point of agent evolution simulation. If system resource meets a basic requirement (i.e., α), the algorithm activates an agent in the searching state within a local station. If the search process finds its goal (e.g., the requested information is found), the goal is achieved. Goal abortion of all agents in a society results in a dangling state of all agents in the same society (including the agent who finds the goal). At the same time, the search result is sent back to the original query station via *Query_Return_URL*. Suppose that the goal can not be achieved in an individual station, the agent is cloned in another station (agent propagation). The *Agent_Clone* algorithm is then used. On the other hand, the agent may be suspended or even killed if the system resource is below the basic requirement (i.e., $Resource\_Available(A, G, X) < \alpha$). In this case, algorithms *Agent_Suspend* is used if the resource available is still feasible for a future resuming of the agent. Otherwise, if the resource is below the minimal requirement, algorithm *Agent..Kill* is used.

Agent Searching Algorithm

```
$Agent\_Search : Agent \times Goal \times Host\_Station$

$\forall A : Agent, G : Goal, X : Host\_Station \bullet$
    $Agent\_Search(A, G, X) \Leftrightarrow$
        $Resource\_Available(A, G, X) \geq \alpha \Rightarrow$
            $[G \in Local\_Search(A, X) \Rightarrow$
                $Abort\_All(A \uparrow Agent\_Society) \wedge$
                $send\_result(X.URL,$
                    $G.Query\_Return\_URL) \wedge$
                $goal\_achieved = TRUE$
            $\vee\ G \notin Local\_Search(A, X) \Rightarrow$
                $Agent\_Clone(A, G,$
                    $A \uparrow Agent\_Society)]$
        $\vee\ Resource\_Available(A, G, X) \geq \beta \Rightarrow$
            $Agent\_Suspend(A, G, X)$
        $\vee\ Resource\_Available(A, G, X) < \beta \Rightarrow$
            $Agent\_Kill(A, G, X)$
```

Agent cloning is achieved by the *Agent_Clone* algorithm. When the cloning process wants to find new stations to broadcast an agent, two implementations can be considered. The first is to collect all URLs of stations found by one search engine. But, considering the network resource available, the implementation may check for the common URLs found by two or more search engines. New URLs are collected by the *Search_For_Stations* algorithm, which is invoked in the agent cloning algorithm. Agent propagation strategy decides the computation efficiency of our model. In this research, we propose three strategies:

- the brute force agent distribution
- the semi-brute force agent distribution, and
- the selective agent distribution.

The first strategy simply clone an agent on a remote station, if the potential station contains information that helps the agent to achieve its goal. The semi-brute force strategy, however, finds another agent on a potential station, and assigns the goal to that agent. The selective approach not only try to find a useful agent, but also check for the goals of that agent. Cloning strategies affect the size of agent societies thus the efficiency of computation.

Agent Cloning Algorithm: the Brute Force Strategy

```
$Agent\_Clone : Agent \times Goal \times Agent\_Society$

$\forall A : Agent, G : Goal, S : Agent\_Society \bullet$
    $Agent\_Clone(A, G, S) \Leftrightarrow$
        $[\forall X : Host\_Station \bullet$
                $X \in Search\_For\_Stations(G) \Rightarrow$
            $(\exists A' : Agent \bullet A' = copy(A) \wedge$
            $X.Agent\_Set = X.Agent\_Set \cup \{ A' \} \wedge$
            $S = S \cup \{ A' \} \wedge$
            $Agent\_Search(A', G, X))]$
        $\vee\ [Search\_For\_Stations(G) = \emptyset \Rightarrow$
            $goal\_achieved = FALSE]$
```

The brute force agent distribution strategy makes a copy of agent A, using the *copy* function, in all stations returned by the *Search_For_Stations* algorithm. Agent set in each station is updated and the society S where agent A belongs is changed. Agent A', a clone of agent A is transmitted to station X for execution.

Agent Cloning Algorithm: the Semi-brute Force Strategy

```
$Agent\_Clone : Agent \times Goal \times Agent\_Society$

$\forall A : Agent, G : Goal, S : Agent\_Society \bullet$
    $Agent\_Clone(A, G, S) \Leftrightarrow$
        $[\forall X : Host\_Station \bullet$
            $X \in Search\_For\_Stations(G) \Rightarrow$
            $[\exists A' : Agent \bullet A' \in X.Agent\_Set \Rightarrow$
                $(A'.Goal\_Set = A'.Goal\_Set \cup$
                        $\{ G \} \wedge$
                    $S = S \cup \{ A' \} \wedge$
                    $Agent\_Search(A', G, X))]]$
        $\vee\ [Search\_For\_Stations(G) = \emptyset \Rightarrow$
            $goal\_achieved = FALSE]$
```

The semi-brute force agent distribution approach is similar to the brute force approach, except that it does not make a copy of the agent but give the goal to an agent on its destination station. The agent which accepts this new goal (i.e., A') is activated for the new goal in its belonging station.

Agent Cloning Algorithm: the Selective Strategy

```
$Agent\_Clone : Agent \times Goal \times Agent\_Society$
---
$\forall A : Agent, G : Goal, S : Agent\_Society \bullet$
    $Agent\_Clone(A, G, S) \Leftrightarrow$
        $[\forall X : Host\_Station \bullet$
            $X \in Search\_For\_Stations(G) \Rightarrow$
            $[\exists A' : Agent \bullet A' \in X.Agent\_Set \Rightarrow$
                $[G \in A'.Goal\_Set \Rightarrow$
                    $S = S \cup A' \uparrow Agent\_Society$
                $\vee G \notin A'.Goal\_Set \Rightarrow$
                    $(A'.Goal\_Set =$
                        $A'.Goal\_Set \cup \{ G \}$
                    $\wedge S = S \cup \{ A' \})]$
                $\wedge Agent\_Search(A', G, X)]$
            $\vee [X.Agent\_Set = \emptyset \Rightarrow$
                $[\exists A'' : Agent \bullet A'' = copy(A) \wedge$
                $X.Agent\_Set = \{ A'' \} \wedge$
                $S = S \cup \{ A'' \} \wedge$
                $Agent\_Search(A'', G, X)]]]$
        $\vee [Search\_For\_Stations(G) = \emptyset$
            $\Rightarrow goal\_achieved = FALSE]$
```

The last approach is more complicate. The selective approach of cloning algorithm must check whether there is another agent in the destination station (i.e., X). If so, the algorithm checks whether the agent (i.e., A') at that station shares the same goal with the agent to be cloned. If two agents share the same goal, there is no need of cloning another copy of agent. Basically, the goal can be computed by the agent at the destination station. In this case, the union of the two societies is necessary (i.e., $S = S \cup A' \uparrow Agent\_Society$). On the other hand, if the two agents do not have a common goal, to save computation resource, we may ask the agent at the destination station to help searching for an additional goal. This case makes a re-organization of the society where the source agent belongs. The result also ensure that the number of agents on the ACN is kept in a minimum. Whether the two agents share the same goal, the *Agent_Search* algorithm is used to search for the goal again. In this case, Agent A' is physically transmitted to station X for execution. When there is no agent running on the destination station, we need to increase the number of agents on the ACN by duplicating an agent on the destination station (i.e., the invocation of $A'' = copy(A)$). The society is reorganized. And the *Agent_Search* algorithm is called again. In the acse that no new station is found by the *Search_For_Stations* algorithm, the goal is not achieved.

The agent search and agent clone algorithms use some auxiliary algorithms, which are discussed as follows. The justification of system resource available depends on agent policy, as defined in $A.Policy$. Agent policy is a set of factors indicated by name tags (e.g., $NETWORK\_BOUND$). The estimation of resources is represented as a real number, which is computed based on $X.Resource$ of station X. Note that, in the algorithm, $w1$ and $w2$ are weights of factors ($w1 + w2 = 1.0$). We only describes some cases of using agent policies. Other cases are possible but omitted. Moreover, we consider the priority of goal G. If the priority is lower than some watermark (i.e., $G.Priority < \theta$), we let $r1$ be a constant less than 1.0. Therefore, resources are reserved for other agents. On the other hand, if the priority is high, we consider the value returned by *Resource_Available* should be high. Thus the potential agent can proceed its computation immediately. The values of θ and ω depend on agent applications.

Auxiliary Algorithms

```
$Resource\_Available : Agent \times Goal \times Host\_Station \rightarrow$
    $REAL$
---
$\forall A : Agent, G : Goal, X : Host\_Station, R : REAL \bullet$
    $\exists w1, w2, r1, r2 : REAL \bullet$
    $Resource\_Available(A, G, X) = R \Leftrightarrow$
        $[NETWORK\_BOUND \in A.Policy \Rightarrow$
            $R = X.Resource.Network$
        $\vee CPU\_BOUND \in A.Policy \Rightarrow$
            $R = X.Resource.CPU$
        $\vee MEMORY\_BOUND \in A.Policy \Rightarrow$
            $R = X.Resource.Memory$
        $\vee CPU\_BOUND \in A.Policy \wedge$
        $MEMORY\_BOUND \in A.Policy \Rightarrow$
            $R = X.Resource.CPU * w1 +$
                $X.Resource.Memory * w2 \wedge$
            $w1 + w2 = 1.0$
        $\vee ...]$
        $\wedge \exists \theta, \omega : Priority \bullet$
            $[G.Priority < \theta \Rightarrow$
                $(R = R * r1 \wedge r1 < 1.0)$
            $\vee G.Priority > \omega \Rightarrow$
                $(R = R * r2 \wedge r2 > 1.0)]$
```

The above algorithms describe how an agent evolves from a state to another. How agents affect each other depends on the system resource available. However, in an ACN, it is possible that agents suspend or even kill each other, as we described in previous sections. The niche overlap graphs of each goal play an important role. We use the *Agent_Suspend* and *Agent_Kill* algorithms to take the niche overlap graphs of a goal (i.e., $niche\_compete(G)$) into consideration. In the *Agent_Suspend* algorithm, if there exists a goal that has a lower priority comparing to the goal of the searching agent, a suspend message is sent to the goal to delay its search (i.e., via $suspend(G' \uparrow Agent)$). The searching agent may be resumed after that since system resources may be released from those goal suspension. In the *Agent_Kill* algorithm, however, a kill message is sent instead (i.e., via $terminate(G' \uparrow Agent)$). The system resource is checked against the minimum requirement β. If resuming is feasible, the *Agent_Search* algorithm in invoked. Otherwise, the system should terminate the searching agent.

$$
\begin{array}{|l}
\mathit{Agent\_Suspend} : \mathit{Agent} \times \mathit{Goal} \times \mathit{Host\_Station} \\
\hline
\forall A : \mathit{Agent}, G : \mathit{Goal}, X : \mathit{Host\_Station} \bullet \\
\qquad \mathit{Agent\_Suspend}(A, G, X) \Leftrightarrow \\
\qquad \exists\, GS : \mathit{Goal\_Set} \bullet \\
\qquad\qquad GS = \mathit{niche\_compete}(G) \\
\qquad\qquad \wedge\ (\forall\, G' : \mathit{Goal} \bullet G' \in GS\ \wedge \\
\qquad\qquad\qquad G'.\mathit{Priority} < G.\mathit{Priority} \Rightarrow \\
\qquad\qquad\qquad\qquad \mathit{suspend}(G' \upharpoonright \mathit{Agent})) \\
\qquad\qquad \wedge\ (\mathit{Resource\_Available}(A, G, X) \geq \beta \Rightarrow \\
\qquad\qquad\qquad \mathit{Agent\_Search}(A, G, X) \\
\qquad\qquad \vee\ \mathit{Resource\_Available}(A, G, X) < \beta \Rightarrow \\
\qquad\qquad\qquad \mathit{suspend}(A))
\end{array}
$$

$$
\begin{array}{|l}
\mathit{Agent\_Kill} : \mathit{Agent} \times \mathit{Goal} \times \mathit{Host\_Station} \\
\hline
\forall A : \mathit{Agent}, G : \mathit{Goal}, X : \mathit{Host\_Station} \bullet \\
\qquad \mathit{Agent\_Kill}(A, G, X) \Leftrightarrow \\
\qquad \exists\, GS : \mathit{Goal\_Set} \bullet \\
\qquad\qquad GS = \mathit{niche\_compete}(G) \\
\qquad\qquad \wedge\ (\forall\, G' : \mathit{Goal} \bullet G' \in GS\ \wedge \\
\qquad\qquad\qquad G'.\mathit{Priority} < G.\mathit{Priority} \Rightarrow \\
\qquad\qquad\qquad\qquad \mathit{terminate}(G' \upharpoonright \mathit{Agent})) \\
\qquad\qquad \wedge\ (\mathit{Resource\_Available}(A, G, X) \geq \beta \Rightarrow \\
\qquad\qquad\qquad \mathit{Agent\_Search}(A, G, X) \\
\qquad\qquad \vee\ \mathit{Resource\_Available}(A, G, X) < \beta \Rightarrow \\
\qquad\qquad\qquad \mathit{terminate}(A))
\end{array}
$$

The other auxiliary algorithms are relatively less complicated. Function *Local_Search* takes as input an agent and a station. It returns a set of goals found by the agent in that station. A *match* predicate is used. This match predicate is application dependent. It could be a search program which locates a key word in a Web page, or a request of information from a user (e.g., a survey questionnaire). The *Abort_All* predicate takes as input an agent society and terminates all agents within that society. The *Search_For_Stations* function takes as input a goal and returns a set of host stations. The stations should be selected depending on the *candidate_station* function, which estimates the possibility of goal achievement in a station. This function can be implemented as a Web search engine which looks for candidate URLs. We have omitted some detailed definitions of the above auxiliary algorithms, as well as some primitive functions which are self-explanatory.

$$
\begin{array}{|l}
\mathit{Local\_Search} : \mathit{Agent} \times \mathit{Host\_Station} \rightarrow \mathit{Goal\_Set} \\
\hline
\forall A : \mathit{Agent}, X : \mathit{Host\_Station}, GS : \mathit{Goal\_Set} \bullet \\
\qquad \mathit{Local\_Search}(A, X) = GS \Leftrightarrow \\
\qquad\qquad GS = \{\ G : \mathit{Goal} \mid G \in A.\mathit{Goal\_Set}\ \wedge \\
\qquad\qquad\qquad \mathit{match}(G.\mathit{Query}, \\
\qquad\qquad\qquad\qquad X.\mathit{Resource}.\mathit{Information})\ \}
\end{array}
$$

$$
\begin{array}{|l}
\mathit{Abort\_All} : \mathit{Agent\_Society} \\
\hline
\forall S : \mathit{Agent\_Society} \bullet \\
\qquad \mathit{Abort\_All}(S) \Leftrightarrow \\
\qquad\qquad \forall A : \mathit{Agent} \bullet A \in S \Rightarrow \mathit{terminate}(A)
\end{array}
$$

$$
\begin{array}{|l}
\mathit{Search\_For\_Stations} : \mathit{Goal} \rightarrow \mathbb{P}\ \mathit{Host\_Station} \\
\hline
\forall G : \mathit{Goal}, X\_\mathit{Set} : \mathbb{P}\ \mathit{Host\_Station} \bullet \\
\qquad \mathit{Search\_For\_Stations}(G) = X\_\mathit{Set} \Leftrightarrow \\
\qquad\qquad X\_\mathit{Set} = \{\ X : \mathit{Host\_Station} \mid \\
\qquad\qquad\qquad \mathit{candidate\_station}(G, X)\ \}
\end{array}
$$

5 Conclusions

Mobile agent based software engineering is interesting. However, in the literature, we did not find any other similar theoretical approach to model what mobile agents should act on the Internet, especially how mobile agents can cooperate and compete. A theoretical computation model for agent evolution was proposed in this paper. Algorithms for the realization of our model were also given.

References

[1] David Kotz, Robert Gray, Saurab Nog, Daniela Rus, Sumit Chawla, and George Cybenko, "Agent Tcl: targeting the needs of mobile computers," IEEE Internet Computing, Vol. 1, No. 4, July 1997, pp. 58 – 67.

[2] S. Krause and T. Magedanz, "Mobile service agents enabling intelligence on demand in telecommunications," in Proceedings of the 1996 IEEE Global Telecommunications Conference, London, UK, 1996, pp 78 – 84.

[3] Sven Krause, Flavio Morais de Assis Silva, and Thomas Magedanz, "MAGNA - a DPE-based platform for mobile agents in electronic service markets," in Proceedings of the 1997 3rd International Symposium on Autonomous Decentralized Systems (ISADS'97), Berlin, Germany, 1997, pp. 93 – 102.

[4] Anselm Lingnau and Oswald Drobnik, "Making mobile agents communicate: a flexible approach," in Proceedings of the 1996 1st Annual Conference on Emerging Technologies and Applications in Communications, Portland, OR, USA, 1996, pp. 180 – 183.

[5] Michael Pazzani and Daniel Billsus, "Learning and Revising User Profiles: The Identification of Interesting Web Sites", Machine Learning, Vol. 27, 1997, pp. 313 – 331.

[6] Ruud Schoonderwoerd, Owen Holland, and Janet Bruten, "Ant-like agents for load balancing in telecommunications networks," in Proceedings of the 1997 1st International Conference on Autonomous Agents, Marina del Rey, California, U.S.A., 1997, pp. 209 – 216.

[7] Joseph Tardo and Luis Valente, "Mobile agent security and telescript," in Proceedings of the 1996 41st IEEE Computer Society International Conference (COMPCON'96), Santa Clara, CA, USA, 1996, pp. 58 – 63.